VILNIAUS UNIVERSITETAS

ALBERTAS GIMBUTAS

NEIŠKILIOJO OPTIMIZAVIMO ALGORITMAS SU NAUJU BIKRITERINIU POTENCIALIŲJŲ SIMPLEKSŲ IŠRINKIMU NAUDOJANT LIPŠICO KONSTANTOS ĮVERTĮ

Daktaro disertacijos santrauka
Fiziniai mokslai, informatika (09P)

Vilnius, 2018

Disertacija rengta 2013–2017 metais Vilniaus universitete.

Mokslinis vadovas

prof. habil. dr. Antanas Žilinskas (Vilniaus universitetas, fiziniai mokslai, informatika – 09P).

Disertacija ginama viešame disertacijos Gynimo tarybos posėdyje:

Pirmininkė

prof. dr. Olga Kurasova (Vilniaus universitetas, fiziniai mokslai, informatika – 09P).

Nariai:

prof. habil. dr. Rimantas Barauskas (Kauno technologijos universitetas, fiziniai mokslai, informatika – 09P),

prof. dr. Dmitry E. Kvasov (Kalabrijos universitetas, Italija, fiziniai mokslai, informatika – 09P),

dr. Remigijus Paulavičius (Vilniaus universitetas, fiziniai mokslai, informatika – 09P),

doc. dr. Dmitrij Šešok (Vilniaus Gedimino technikos universitetas, technologijos mokslai, informatikos inžinerija – 07T).

Disertacija bus ginama viešame Gynimo tarybos posėdyje 2018 m. rugsėjo 21 d. 12 val. Vilniaus universiteto Duomenų mokslo ir skaitmeninių technologijų instituto 203 auditorijoje. Adresas: Akademijos g. 4, LT-04812 Vilnius, Lietuva.

Disertacijos santrauka išsiuntinėta 2018 m. rugpjūčio 20 d.
Disertaciją galima peržiūrėti Vilniaus universiteto bibliotekoje ir VU interneto svetainėje adresu: *www.vu.lt/lt/naujienos/ivykiu-kalendorius*

VILNIUS UNIVERSITY

ALBERTAS GIMBUTAS

NONCONVEX OPTIMIZATION ALGORITHM WITH A NEW BI-CRITERIA SELECTION OF POTENTIAL SIMPLICES USING AN ESTIMATE OF LIPSCHITZ CONSTANT

Summary of Doctoral Dissertation
Physical sciences, Informatics (09P)

Vilnius, 2018

The dissertation work was carried out at Vilnius University from 2013 to 2017.

Scientific supervisor

Prof. Habil. Dr. Antanas Žilinskas (Vilnius University, Physical Sciences, Informatics – 09P).

The dissertation is defended at the Dissertation Defence Counsil:

Chairman

Prof. Dr. Olga Kurasova (Vilnius University, Physical Sciences, Informatics – 09P).

Members:

Prof. Habil. Dr. Rimantas Barauskas (Kaunas University of Technology, Physical Sciences, Informatics – 09P),

Prof. Dr. Dmitry E. Kvasov (University of Calabria, Italy, Physical Sciences, Informatics – 09P),

Dr. Remigijus Paulavičius (Vilnius University, Physical Sciences, Informatics – 09P),

Assoc. Prof. Dr. Dmitrij Šešok (Vilnius Gediminas Technical University, Technological Sciences, Informatics Engineering – 07T).

The dissertation will be defended at the public meeting of the Council in the auditorium 203 of the Institute of Data Science and Digital Technologies of Vilnius University on the 21st of September, 2018 at 12:00.
Address: Akademijos str. 4, LT–04812 Vilnius, Lithuania.

The summary of the doctoral dissertation was distributed on the 20th of August, 2018.
The dissertation is available at the library of Vilnius University.

Santrauka

Disertacijoje nagrinėjami DIRECT (DIviding RECTangles) tipo Lipšico tikslo funkcijos modeliais su nežinoma Lipšico konstanta pagrįsti optimizavimo algoritmai, kurie yra dažnai taikomi praktinių „juodosios dėžės“ tipo uždavinių sprendimui. To pagrindu išsikeltas pagrindinis disertacijos tikslas – pasiūlyti globaliojo optimizavimo Lipšico klasės su nežinoma Lipšico konstanta algoritmą, efektyviai išnaudojantį potencialiai brangius funkcijų skaičiavimus. Nagrinėjamai algoritmų klasei yra pasiūlytas naujas simpleksinis algoritmas LIBRE (LIpschitz Bound Rough Estimation), paremtas ankstesniais DISIMPL (DIviding SIMPLices) algoritmais. Pasiūlytojo algoritmo naujumas tas, kad naudojamas vienas Lipšico konstantos įvertis vietoje aibės Lipšico konstantų, parenkant potencialiuosius simpleksus dalinimui. Atlikta eksperimentinė analizė parodė pasiūlytojo algoritmo konkurencingumą su kitais geriausiais šios klasės algoritmais. Taip pat, yra eksperimentiškai ištiriamos įvairios LIBRE algoritmo modifikacijos; parodoma surogatinių Lipšico rėžių griežtumo įtaka pasiūlytojo algoritmo efektyvumui. Be to, yra pasiūlyta strategija, kaip Lipšico globaliojo optimizavimo algoritmus apibendrinti daugelio kriterijų optimizavimo uždavinių sprendimui. LIBRE algoritmas apibendrintas daugiakriteriniams uždaviniams, pritaikius pasiūlytąją strategiją, ir eksperimentiškai ištirtas.

Turinys

Žymėjimai

D	leistinoji sritis
X	aibė taškų, kuriuose įvertinta tikslo funkcijos reikšmė
Y	tikslo funkcijos įvertinimų aibė
Y^{Pareto}	Pareto fronto diskreti aproksimacija
$\overline{D}$	vienetinis hiperkubas, gautas normalizavus D
d	leistinosios srities dimensijų skaičius, t. y. $D \in \mathbb{R}^d$
n	tikslo funkcijų skaičius
m	tikslo funkcijos įvertinimų skaičius tam tikroje algoritmo iteracijoje
$\boldsymbol{x}$	vektorius
$f(\cdot)$	tikslo funkcija
$\boldsymbol{f}$	tikslo funkcijų vektorius
f_{min}^m	mažiausia rasta tikslo funkcijos reikšmė po m įvertinimų
$g(\cdot)$	Lipšico apatinių rėžių funkcija (Lipšico minorantas) po tikslo funkcija
$\boldsymbol{g}$	Lipšico minorantų vektorius (kiekvienai tikslo funkcijai po vieną)
$\boldsymbol{x}^*$	globaliojo minimumo taškas
f^*	globalusis minimumas
$\mathbb{R}$	realiųjų skaičių aibė
L_p	Lipšico konstanta p-normos atstumo atžvilgiu
L	Lipšico konstanta Euklidinio atstumo atžvilgiu, taip pat žymima kaip L_2
$\widetilde{L}$	Lipšico konstantos L_2 įvertis
S	simpleksų aibė
$V(\cdot)$	viršūnių aibė
$\Delta(\cdot)$	diametras
$\lvert S \rvert$	aibės S dydis, t. y. elementų skaičius aibėje S
$\lVert \cdot \rVert_p$	atstumas p-normos atžvilgiu, t. y. $\lVert \boldsymbol{x} \rVert_p = \left(\sum_{i=1}^d \lvert x_i \rvert^p \right)^{1/p}$
$\lVert \cdot \rVert$	Euklidinis atstumas, taip pat žymimas ir kaip $\lVert \cdot \rVert_2$

1 Įvadas

1.1 Tyrimų sritis

Daugelyje inžinerijos sričių iškyla poreikis surasti parametrų kombinaciją paieškos srityje, atitinkančią tam tikros tikslo funkcijos globalųjį optimumą. Globaliai optimalaus sprendinio pranašumas lokaliai optimalių sprendinių atžvilgiu yra akivaizdus, tačiau tokio sprendinio radimas yra daug sudėtingesnis uždavinys, reikalaujantis papildomų pastangų ir resursų. Šio uždavinio sprendimo būdus nagrinėja globaliojo optimizavimo tyrimų sritis, apimanti tiek susijusią teoriją, tiek optimizavimo algoritmų realizavimo klausimus. Dažnai taikomuosiuose uždaviniuose tikslo funkcija yra juodoji dėžė, t. y. jos analitinė išraiška nežinoma, o norint apskaičiuoti funkcijos reikšmę, būtina įvykdyti brangų skaitinį eksperimentą. Dėl to bendruoju atveju griežtos prielaidos apie tikslo funkcijos savybes neturėtų būti daromos ir į keleto lokaliųjų minimumų egzistavimo galimybę turėtų būti atsižvelgta. Dėl didelės kainos, susijusios su kiekvienos funkcijos reikšmės apskaičiavimu, svarbu nustatyti globalaus minimumo įvertį kuo mažesnėmis sąnaudomis, t. y. atlikus kuo mažiau tikslo funkcijos įvertinimų. Dėl to svarbu kurti naujus vis didesnio efektyvumo globaliojo optimizavimo algoritmus.

Viena iš paplitusių prielaidų apie tikslo funkciją yra ta, kad aprėžtas kintamųjų pokytis lemia aprėžtą tikslo funkcijos reikšmės pokytį. Matematiškai ši prielaida įforminama kaip Lipšico sąlyga tikslo funkcijai. Prielaida pasitarnauja algoritmų kūrimui ir teoriniam tyrimui, pavyzdžiui, tampa įmanoma nustatyti algoritmų konvergavimo savybes bei garantuoti gauto globaliojo minimumo artinio tikslumą. Be to, Lipšico sąlyga dažniausiai naudojama deterministiniuose algoritmuose, kurie pranašesni už stochastinius algoritmus tuo, kad rezultatui sužinoti pakanka juos įvykdyti vieną kartą. Pagrindinė šios klasės algoritmų problema realistiniuose scenarijuose yra nežinoma Lipšico konstanta. Ją galima ignoruoti supaprastinimo sumetimais laikant, kad konstanta žinoma, arba spręsti pasitelkiant įvarias konstantos įvertinimo strategijas. Vieno kintamojo Lipšico globalusis optimizavimas yra gerai ištirtas tiek teoriniu, tiek praktiniu aspektais. Kita vertus, daugelio kintamųjų atveju globaliojo optimizavimo uždavinys tampa kur kas sudėtingesnis, dėl to tyrimai vyksta toliau, pasitelkiant šakų ir rėžių algoritminės schemos variacijas bei Lipšico konstantos kitimą aibėje leistinų konstantų.

1.2 Darbo aktualumas

Daugelyje taikomųjų optimizavimo uždavinių tikslo funkcijos apskaičiavimas yra brangus, nes apima ilgai trunkančio skaitinio eksperimento vykdymą. Dėl to svarbu siekti didesnio optimizavimo algoritmų efektyvumo tikslo funkcijos apskaičiavimų skaičiaus atžvilgiu. Buvo nustatyta, kad šiuo atžvilgiu Lipšico optimizavimo algoritmai, naudojantys adaptyvų Lipšico konstantos įvertį, yra efektyvūs. Tačiau Lipšico konstantos įverčio naudojimas neleidžia garantuoti gautojo sprendinio tikslumo. Kita vertus, algoritmų klasė, naudojanti aibę leistinų Lipšico konstantų vietoje vieno jos įverčio, leidžia garantuoti, kad globalusis minimumas bus aproksimuotas reikiamu tikslumu apskaičiavus funkcijos reikšmę baigtinį kartų skaičių. Dėl to verta ieškoti šiuos skirtingus algoritmus pagrindžiančių idėjų derinimo būdų, siekiant kurti efektyvius algoritmus, garantuojančius ir sprendinio tikslumą.

Atrodo intuityvu, kad informacija apie tikslo funkciją yra tikslesnė, kai jai apskaičiuojami griežtesni Lipšico rėžiai. Kuo geriau tikslo funkcijos modelis atitinka tikrąją situaciją, tuo patikimesnė informacija prieinama optimizavimo algoritmui renkantis naują funkcijos apskaičiavimo tašką. Dėl to svarbu ištirti, kokį poveikį optimizavimo efektyvumui turi įvairaus griežtumo Lipšico rėžiai.

Daugiakriterinio optimizavimo uždaviniai dažniausiai sprendžiami pasitelkiant metaeuristinius algoritmus. Šiai kategorijai priklausantys algoritmai yra randomizuoti, dėl to juos tenka įvykdyti daugelį kartų, be to, nesuteikiama garantija, kad jais gauti sprendiniai iš tiesų yra Pareto optimalūs tam tikru tikslumu. Kita vertus, literatūroje buvo pademonstruota, kad deterministiniai algoritmai yra efektyvesni už metaeuristinius sprendžiant paprastus optimizavimo uždavinius. Tai skatina toliau tirti deterministinių algoritmų potencialą ir plėsti aibę jais sprendžiamų uždavinių, siekiant deterministinių algoritmų efektyvumą priartinti prie metaeuristinių, tuo pat metu išvengiant metaeuristinių algoritmų trūkumų.

1.3 Darbo tikslai ir uždaviniai

Bendriausias šio darbo tikslas yra pasiūlyti tikslo funkcijos apskaičiavimų skaičiaus atžvilgiu efektyvius Lipšico optimizavimo algoritmus, nenaudojančius konkrečios Lipšico konstantos.

Iškelti šie tikslai:

1. Pasiūlyti algoritmą, kombinuojantį DIRECT tipo algoritmų ir Lipšico konstantos įverčius naudojančių algoritmų teigiamus aspektus siekiant efektyvesnio veikimo tikslo funkcijos apskaičiavimų skaičiaus atžvilgiu.
2. Ištirti surogatinių Lipšico rėžių griežtinimo įtaką sudėtingų daugiaekstremalių tikslo funkcijų optimizavimui.
3. Pasiūlyti Lipšico algoritmų adaptavimo daugiakriteriniams optimizavimo uždaviniams būdą, kad gauto optimizavimo proceso efektyvumas prilygtų šiuolaikinių genetinių algoritmų efektyvumui.

Tikslams pasiekti, suformuluoti šie uždaviniai:

1. Identifikuoti DIRECT tipo simpleksinių Lipšico optimizavimo algoritmų teigiamus aspektus ir pasiūlyti algoritmo modifikaciją, išsaugančią šiuos aspektus bei naudojančią adaptyviai randamą Lipšico konstantos įvertį, kad būtų gautas geresnių savybių optimizavimo algoritmas.
2. Eksperimentiškai palyginti pasiūlytųjų ir kitų populiarių algoritmų veikimą panaudojant sudėtingas dirbtiniu būdu sugeneruotas testines funkcijas.
3. Eksperimentiškai palyginti skirtingo griežtumo surogatinių Lipšico rėžių simplekso viduje apskaičiavimo strategijų įtaką optimizavimo algoritmų veikimui.
4. Apibendrinti pasiūlytąjį algoritmą daugiakriteriniams uždaviniams ir atlikti eksperimentinį palyginimą su šiuolaikiniais genetiniais algoritmais.

1.4 Darbo naujumas ir rezultatai

Vieno kintamojo Lipšico globaliosios optimizacijos su simpleksiniais apibrėžimo srities posričiais kontekste buvo pasiūlyta sukombinuoti du simpleksų išrinkimo tolimesniam dalijimui kriterijus. Pirmasis kriterijus yra minimali surogantinių Lipšico rėžių reikšmė simplekso viduje, antrasis – simplekso diametras. Ekperimentiškai pademonstruota, kad šia idėja pagrįstas algoritmas blogiausiu atveju veikia geriau nei kiti nagrinėti algoritmai, optimizuojant sunkias tikslo funkcijas.

Išmėgintos kelios surogatinių Lipšico rėžių skaičiavimo strategijos pasiūlytajam algoritmui. Strategijos tarpusavyje skiriasi gaunamų Lipšico rėžių griežtumu ir skaičiavimų sudėtingumu. Eksperimentiškai nustatyta, kad griežtesni Lipšico rėžiai nebūtinai nulemia didesnį optimizavimo algoritmo efektyvumą tikslo funkcijos įvertinimų skaičiaus atžvilgiu.

Pasiūlytas būdas apibendrinti daugiakriteriniam atvejui Lipšico optimizavimo algoritmus, skirtus vieno kriterijaus uždaviniams bei pagrįstus rekursyviu apibrėžimo srities posričių dalijimu. Apibendrinimas atliktas remiantis ankstesnėmis vieno žingsnio blogiausiu atveju optimalaus algoritmo idėjomis. Šis apibendrinimas pritaikytas darbe pasiūlytam vieno kriterijaus optimizavimo algoritmui ir eksperimentiškai pademonstruota, kad gautasis daugiakriterinio optimizavimo algoritmas mažos dimensijos uždaviniams veikia panašiai kaip ir populiarus genetinis algoritmas.

1.5 Ginamieji teiginiai

1. Pasiūlytasis globaliojo optimizavimo algoritmas Libre yra blogiausiu atveju efektyvesnis tikslo funkcijos įvertinimų skaičiaus atžvilgiu sudėtingiems globaliojo optimizavimo uždaviniams nei kitos populiarios alternatyvos.

2. Tikslesnių surogatinių Lipšico rėžių naudojimas ne visada padidina Direct tipo optimizavimo algoritmo efektyvumą tikslo funkcijos įvertinimų skaičiaus atžvilgiu.

3. Daugiakriterinė pasiūlytojo Libre algoritmo versija yra panašaus efektyvumo tikslo funkcijų įvertinimų skaičiaus atžvilgiu kaip ir populiarus genetinis algoritmas NSGA-II sprendžiant žemo dimensiškumo daugiakriterinius optimizavimo uždavinius.

2 Lipšico tolydžių funkcijų optimizacijos apžvalga

2.1 Globaliojo optimizavimo uždavinys

Sakykime, $f(x) : D \subset \mathbb{R}^d \to \mathbb{R}$. Globaliojo optimizavimo uždavinys yra rasti

$$\min_{x \in D} f(x). \tag{1}$$

Aibė D vadinama apibrėžimo sritimi, o $f(x)$ – tikslo funkcija. Kai funkcijos reikšmes galima apskaičiuoti, tačiau jos analitinė išraiška nežinoma, $f(x)$ vadinama juodosios dėžės funkcija. Uždavinio (1) sudėtingumas kyla dėl aplinkybės, kad $f(x)$ gali turėti daugiau kaip vieną lokalųjį minimumą ir dėl to lokaliojo optimizavimo algoritmai netinkami jam spręsti. Uždavinio sprendinių aibė žymima $X^* = \{x^* : f(x^*) = \min_{x \in D} f(x)\}$, kur $x^* \in X^*$ vadinamas globalaus minimumo tašku, o reikšmė $f^* = f(x^*)$, $x^* \in X^*$, yra globalusis minimumas. Kartais pakanka rasti apytikslę f^* reikšmę, t. y. $\widehat{x}^* \in D : f(\widehat{x}^*) \leqslant f^* + \epsilon, \epsilon > 0$. Literatūroje nagrinėti įvairūs konkretūs (1) uždavinio atvejai [31], besiskiriantys prielaidomis apie D ir f, ir leidžiantys jį spręsti panaudojant specifinius algoritmus.

Sakoma, kad $f(x)$ tenkina Lipšico sąlygą, kai:

$$|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leqslant L\|\mathbf{x}_1 - \mathbf{x}_2\|, \forall \mathbf{x}_1, \mathbf{x}_2 \in D, \tag{2}$$

čia L vadinama Lipšico konstanta, o $\|\cdot\|$ dažniausiai yra Euklidinė norma. Prielaida (2) labai bendra ir realistiška, taip pat ypač naudinga teoriniams algoritmų tyrimams bei konstruojant naujus optimizavimo algoritmus. Vis dėlto konstanta L dažniausiai nežinoma ir tai yra pagrindinis prielaidos naudojimo trūkumas.

2.2 Lipšico globaliojo optimizavimo algoritmai

Turėdami tikslo funkcijos įvertinimo taškų seką $x_i, i = 1, \ldots, m$, $x_i \in D$, pažymėkime geriausią žinomą funkcijos reikšmę $f^m_{min} = \min_{i=1,\ldots,m} f(x_i)$. Globaliojo optimizavimo algoritmas yra tokią seką generuojanti procedūra, užtikrinanti, kad $f^m_{min} \to f^*, m \to \infty$. Globaliojo minimumo reikšmės ir taško aproksimacijomis laikomi f^m_{min} ir $x_j, j \in$

$\{1,\ldots,m\}$, $f(x_j) = f^m_{min}$. Kai $D \subset \mathbb{R}$ ir taškai $x_i, i = 1,\ldots,m$, surikiuojami didėjimo tvarka, jie žymimi x^m_i ir jiems teisinga $x^m_1 \leqslant x^m_2 \leqslant \cdots \leqslant x^m_m$.

Globaliojo optimizavimo algoritmai, naudojantys (2) prielaidą, gali būti klasifikuojami pagal Lipšico konstantos kilmę [35]. Dalis algoritmų naudoja Lipšico konstantą kaip algoritmui pateikiamą žinomą parametrą; antroji dalis adaptyviai skaičiuoja lokalų arba globalų konstantos įvertį, o trečioji dalis naudoja aibę leistinų konstantų vietoj vienos fiksuotos reikšmės. Įvairūs Lipšico globaliojo optimizavimo algoritmai plačiai aptarti darbuose [14, 16, 31, 34, 41].

2.2.1 Lipšico konstanta yra žinoma

Pirmiausia aptarsime algoritmus, kuriuose daroma praktikoje retai teisinga prielaida, kad Lipšico konstanta yra žinoma. Pasyvus tolygaus apibrėžimo srities padengimo algoritmas garantuoja sprendinį mažiausiomis sąnaudomis blogiausios tikslo funkcijos atveju. Vieno kintamojo atveju egzistuoja ir geriausias įmanomas, tačiau tik teoriškai realizuojamas, adaptyvus algoritmas [3, 4], reikalaujantis mažiausio tikslo funkcijos įvertinimų skaičiaus reikiamo tikslumo sprendiniui nustatyti. Daug tyrėjų dėmesio sulaukė Pijavskij-Shubert algoritmas [30, 39], optimalus viename žingsnyje blogiausiu atveju [17]. Literatūroje pasiūlyta įvairių strategijų apibendrinti vienmačius optimizavimo algoritmus daugiamačiam atvejui, pvz., naudojant Peano kreives [2, 40] arba Pijavskij-Shubert algoritmo idėjas [18, 25]. Taip pat populiarūs šakų ir rėžių algoritmai [15] su stačiakampiais [33] ir simpleksiniais posričiais [28].

2.2.2 Lipšico konstanta yra įvertinama

Kadangi Lipšico konstanta retai yra žinoma, literatūroje pasiūlyta naudoti adaptyvų jos įvertį. Kai kuriuose algoritmuose naudojamas vienas globalus L įvertis, taikomas visai apibrėžimo sričiai D [16, 31, 41]. Nors šis metodas labiau pagrįstas nei prielaida apie žinomą Lipšico konstantą, globalus jos įvertis gali būti pernelyg bendras ir neatitikti tikrojo funkcijos kitimo greičio skirtinguose D posričiuose. Šiam neatitikimui likviduoti pasiūlyti vadinamieji lokaliojo derinimo algoritmai [20, 32, 38, 41], leidžiantys subalansuoti lokalią ir globalią informaciją apie tikslo funkciją algoritmo veikimo metu.

2.2.3 Lipšico konstanta parenkama iš aibės galimų reikšmių

Idėja vietoje fiksuotos Lipšico konstantos ar jos įverčio naudoti iškart aibę galimų jos reikšmių pirmąkart pateikta DIRECT algoritme [19]. Ji įgalino efektyviai subalansuoti lokalią ir globalią paiešką, be to, algoritmo pritaikomumas juodosios dėžės uždaviniams ir geras veikimas lėmė jo taikymą inžinerijoje. DIRECT algoritmas pagrįstas rekursyviu apibrėžimo srities dalijimu hiperstačiakampiais, kurių dydis ir apatinio Lipšico rėžio minimumas naudojamas hiperstačiamapiams išrinkti. Kaip algoritmo trūkumas gali būti paminėtas greitas globaliojo minimumo traukos srities nustatymas ir tolimesnis lėtas konvergavimas jo link. Šiam trūkumui sušvelninti pasiūlyta daug modifikacijų, pavyzdžiui, pakeičiant hiperstačiakampių išrinkimo kriterijų [1, 9]. Taip pat naudoti ir simpleksiniai apibrėžimo srities posričiai vietoje hiperstačiakampių, pvz., algoritmuose DISIMPL-C and DISIMPL-V [29]. Kituose algoritmuose pasiūlytos hiperstačiakampių nufiltravimo schemos, taikant DIRECT tipo optimizacijos algoritmą mažesnei jų aibei ir taip padidinant tokio kelių lygmenų algoritmo efektyvumą [23, 24, 27, 36]. Įvarios kitos modifikacijos pateikiamos darbuose [13, 21, 22, 26].

3 Pasiūlytieji algoritmai ir jų eksperimentinis tyrimas

3.1 Globaliojo optimizavimo algoritmas, naudojantis globalųjį Lipšico konstantos įvertį

Praktikoje globalaus optimizavimo uždaviniams Lipšico konstanta dažniausiai yra nežinoma. Šiems uždaviniams spręsti dažniausiai naudojamas Lipšico optimizacijos algoritmas yra DIRECT ir jo modifikacijos.

Pirmasis DIRECT algoritmo privalumas yra tas, kad jame kiekvienos iteracijos metu yra dalinimui parenkami skirtingo dydžio posritis ir taip kombinuojama globalioji ir lokalioji paieškos. Be to, yra garantuojama, kad globaliojo minimumo taškas bus rastas, kai funkcijų įvertinų skaičius artėja į begalybę.

Antrasis privalumas yra tas, jog naudojami paprasti Lipšico rėžiai, kurie sudaromi naudojant tik vieną posrities centre esantį tašką. Dėl to reikia labai mažai skaitinių

skaičiavimų, kad būtų randamas apatinių Lipšico rėžių minimumas, nes jis gali būti randamas analitiškai.

Kaip bebūtų, DIRECT algoritmas turi kelis trūkumus. Pavyzdžiui, Lipšico konstantos įverčiai yra parenkami iš labai plačios aibės ($[0, \infty)$). Dėl to gali būti naudojami tikslesni įverčiai. Antra, DIRECT tipo algoritmai dažnai išnaudoja per didelį funkcijų įvertinimų skaičių neoptimalių lokaliųjų minimumų tyrinėjimui, taip atidedant globaliojo minimumo taško atradimą.

Šiame skyriuje yra pasiūlomas Lipšico optimizavimo algoritmas, skirtas globaliojo optimizavimo uždaviniams aprėžtoje leistinojoje srityje, kai Lipšico konstanta nežinoma.

Idėja pasirinkti ir dalinti skirtingo dydžio posritis yra perimta iš DIRECT tipo algoritmų. Pagrindiniai skirtumai tarp pasiūlytojo ir DIRECT algoritmo yra:

1. vienas Lipšico konstantos įvertis yra naudojamas kiekvienoje iteracijoje vietoje Lipšico konstantų aibės;
2. yra naudojamas dalinimas simpleksais, o ne stačiakampiais, identiškai kaip ir DISIMPL-V ir GB-DISIMPL-V algoritmuose.
3. pasiūlytasis algoritmas turi vieną parametrą $\alpha \in [0, \infty)$, kuris leidžia pasirinkti paieškos globalumą. Kuo didesnė pasirenkama α reikšmė, tuo globaliau orientuota paieška gaunama (efektyviausias veikimas gaunamas, kai $\alpha \in [0, 1]$).

Taip pat, yra pasiūlomos ir skaitiškai palyginamos kelios pasiūlytojo algoritmo modifikacijos. Pirmiausia, eksperimentiškai palyginami skirtingi būdai apibrėžti surogatinius Lipšico rėžius. Antra, eksperimentiškai palyginamos skirtingos strategijos pasirinkti posritis dalinimui. Galiausiai, palyginamos skirtingos strategijos rasti Lipšico konstantos įvertį.

3.1.1 LIBRE algoritmo aprašymas

Pirmiausia LIBRE algoritme leistinoji sritis yra konvertuojama į vienetinį hiperkubą; Tolimesniuose algoritmo žingsniuose daroma prielaida, kad leistinoji sritis yra padengta nepersidengiančių simpleksų aibe. Pradinis padengimas simpleksais gaunamas pritaikius standartinį kobinatorinį viršūnių trianguliavimo algoritmą [28], skirtą

padengti vienetinį hiperkubą simpleskais. Pradiniame padengime gaunami $d!$ simpleksai, kurie sudaryti iš hiperkubo viršūnių $v_i, i = 1, ..., 2^d$. Tikslo funkcijos reikšmės apskaičiuojamos visose vienetinio hiperkubo viršūnėse. Toliau, panaudojant gautąsias tikslo funkcijos reikšmes, yra apskaičiuojama pradinė Lipšico konstantos įverčio reikšmė $\widetilde{L}$, kuri yra gaunama kaip maksimumas

$$|f(\boldsymbol{v}_i) - f(\boldsymbol{v}_j)|/||\boldsymbol{v}_i - \boldsymbol{v}_j||, \ i \neq j. \tag{3}$$

Dalinimui parenkami simpleksai pritaikius du kriterijus. Pirmasis kriterijus yra simplekso surogatinių Lipšico apatinių rėžių mažiausios reikšmės aproksimacija. Surogatiniai Lipšico apatiniai rėžiai simpleksui S_i yra apibrėžimai naudojant mažiausią funkcijos reikšmę S_i viršūnėse ir einamąjį Lipšico konstantos įvertį $\widetilde{L}_k$:

$$G(S_i, \widetilde{L}_k) = \min_{\boldsymbol{v}_j \in V(S_i)} f(\boldsymbol{v}_j) - \widetilde{L}_k \Delta(S_i)\alpha, \tag{4}$$

čia $V(S_i)$ yra S_i viršūnių aibė, $\Delta(S_i)$ yra S_i diametras ir $\alpha \geqslant 0$ yra pasiūlytojo algoritmo parametras, kuriuo galima reguliuoti paieškos globalumą. Didesnės α reikšmės sąlygoja globalesnę paiešką, o atveju, kai $\alpha = 0$, gaunamas globalaus optimizavimo algoritmas DISIMPL-V [29], kuris pasižymi pernelyg lokalia paieška.

Antrasis parinkimo kriterijus yra simplekso diametras: $\Delta(S_i)$.

Simpleksų aibės parinkimas dalinimui einamojoje iteracijoje gali būti interpretuojamas kaip dviejų kriterijų optimizavimo uždavinys, kai leistinoji sritis susideda iš diskrečios simpleksų aibės, o kriterijai yra $G(\cdot)$ ir $\Delta(\cdot)$ bei siekiama minimizuoti $G(\cdot)$ ir maksimizuoti $\Delta(\cdot)$:

$$\min_{S_i \in S}(G(S_i), -\Delta(S_i)). \tag{5}$$

Simpleksų aibė, atitinkanti šio uždavinio Pareto optimalius sprendinius, yra parenkama dalinimui, kad būtų užtikrinamas geriausias kompromisas tarp (4) ir $\Delta(\cdot)$ kriterijų bei kad parinktųjų simpleksų aibė būtų ne per didelė.

Parinktieji simpleksai yra padalinami pusiau padalinant ilgiausią kraštinę vidurio taške. Atliekami tikslo funkcijos įvertinimai naujuose taškuose, kurie buvo gauti dalinant parinktuosius simpleksus. Tikslo funkcijos reikšmės yra saugomos subalansuoto medžio duomenų struktūroje ir nėra perskaičiuojamos pakartotinai. T. y. tikslo funkcijos

reikšmė paimama iš duomenų struktūros, jeigu reikiamame taške ji jau buvo apskaičiuota ankstesnėje iteracijoje.

Pasiūlytojo algoritmo konvergavimas yra įrodomas fakto, kad kiekvienos pasiūlytojo algoritmo iteracijos metu yra padalinamas simpleksas su ilgiausiu diametru. Toks simpleksas yra parenkamas kiekvienoje iteracijoje, nes jis neišvengiamai priklauso atraminei Pareto optimalių sprendinių aibei: jo dviejų dimensijų vektorius, susidedantis iš $G(\cdot)$ ir $\Delta(\cdot)$, neabejotinai priklauso Pareto frontui, nes turi geriausią vieno iš kriterijų reikšmę. Įvertinti konvergavimo greitį yra sudėtinga. Tuo pat metu algoritmo veikimą galima įvertinti eksperimentiškai, kas ir yra padaroma kitame skyriuje.

Algorithm 1: LIBRE – pasiūlytasis globaliojo optimizavimo algoritmas

Input $\boldsymbol{l}$ – vektorius, susidedantis iš apatinių leistinosios srities rėžių, $\boldsymbol{u}$ – vektorius susidedantis iš viršutinių leistinosios srities rėžių, $\boldsymbol{f}(\cdot)$ – vektorius susidedantis iš tikslo funkcijų, M_{max} – maksimalus funkcijų įvertinimų skaičius.
Function *LIBRE*($\boldsymbol{l}$, $\boldsymbol{u}$, $\boldsymbol{f}$, M_{max})*:*
  Konvertuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.
  Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \left\{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\right\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.
  Apskaičiuoti $\{f(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikali virn $S, i = 1, ..., 2^d\}$. Nustatyti $m = 2^d, \widetilde{L} = 0$.
  while *sustojimo sąlyga nėra patenkinta* **and** $m < M_{max}$ **do**
    foreach $S_l \in S$ **do**
      Rasti $\widetilde{L}_l = \max\left\{\frac{|f(\boldsymbol{v}_i) - f(\boldsymbol{v}_j)|}{||\boldsymbol{v}_i - \boldsymbol{v}_j||} : \boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_l), \boldsymbol{v}_i \neq \boldsymbol{v}_j\right\}$.
    Atnaujinti $\widetilde{L} = \max\left\{\widetilde{L}\right\} \cup \left\{\widetilde{L}_l : S_l \in S\right\}$.
    foreach $S_l \in S$ **do**
      find $G(S_l, \widetilde{L}) = \min_{\boldsymbol{v}_i \in V(S_l)} f(\boldsymbol{v}_i) - \widetilde{L}\Delta(S_l)\alpha$
    Parinkti aibę simpleksų dalinimui:
      $P = \{S_i : S_i \in$
    S, S_i priklauso iskiliesiems Pareto optimaliems (5) sprendiniams$\}$
    foreach $S_l \in P$ **do**
      Padalinti S_l į du naujus simpleksus S_l^1, S_l^2, pridedant viršūnę $\boldsymbol{v}$ ilgiausios S_l briaunos vidurio taške.
      Atnaujinti $S = S \backslash \left\{S_l\right\} \cup \left\{S_l^1, S_l^2\right\}$. If $\boldsymbol{v}$ is a new vertex, evaluate $f(\boldsymbol{v})$ and set $m = m + 1$.
  return f_{min}

Pasiūlytasis algoritmas buvo pavadintas LIBRE (LIpschitz Bounds Rough Estimation), nes surogatinių Lipšico rėžių aproksimacija (4) yra sudaroma naudojant tik vieną vir-

1 lentelė: GKLS testinių funkcijų klasių parametrai

Klasė	Sudėtingumas	d	Minimumų skaičius	f^*	ρ	r	δ
1	Paprasta	2	10	-1	0.90	0.2	10^{-4}
2	Sunki	2	10	-1	0.90	0.1	10^{-4}
3	Paprasta	3	10	-1	0.66	0.2	10^{-6}
4	Sunki	3	10	-1	0.90	0.2	10^{-6}
5	Paprasta	4	10	-1	0.66	0.2	10^{-6}
6	Sunki	4	10	-1	0.90	0.2	10^{-6}
7	Paprasta	5	10	-1	0.66	0.3	10^{-7}
8	Sunki	5	10	-1	0.66	0.2	10^{-7}

šūnę su mažiausia funkcijos reikšme ir dar jie apibrėžiami naudojant Lipšico konstantos įvertį. Pasiūlytojo algoritmo pseudokodas yra pateiktas 1 algoritme. Šis algoritmas buvo realizuotas C++ programavimo kalba, vengiant pasikartojančių skaičiavimų, pavyzdžiui, visos aproksimacijos (4) buvo atnaujinamos tik tada, kai pasikeisdavo Lipšico konstantos įverčio $\widetilde{L}$ reikšmė.

3.1.2 Eksperimentinis tyrimas

Šis algoritmas buvo kuriamas siekiant spręsti sunkias optimizavimo problemas, kai lokalieji minimumai yra kaip spygliai lėtai besikeičiančiame paviršiuje. Taigi, jo veikimas buvo testuotas naudojant testinių funkcijų generatorių GKLS [10], kuriuo galima sugeneruoti norimų charakteristikų funkcijas. Mūsų eksperimentuose buvo naudojamos tokios pačios testinių funkcijų klasės kaip ir kituose straipsniuose [27, 29, 37], t. y. atsitiktinai sugeneruotos testinės funkcijos su parametrų reikšmėmis, pateiktomis 1 lentelėje. Šios funkcijos yra sudarytos kaip (santykinai plokščias) foninis hiper paraboloidas su pridėtais „spygliais“, kurie apibrėžti polinomais. Šių uždavinių sudėtingumas priklauso nuo globaliojo minimumo taško traukos srities dydžio ir nuo atstumo tarp globaliojo ir foninio paraboloido minimumų.

Šių parametrų reikšmės sutampa visoms testinių funkcijų klasėms: lokalių minimumų skaičius yra lygus 10, globaliojo minimumo reikšmė yra -1. Kiekvienai testinių funkcijų klasei specifiniai parametrai pateikti 1 lentelėje. Naudojami tokie žymėjimai: ρ – atstumas tarp globaliojo minimumo ir foninio paraboloido minimumo taškų, r – globaliojo minimumo taško traukos srities diametras. Dar kartą paminėsime, kad d

2 lentelė: α parametro įtaka pasiūlytojo algoritmo veikimui, t. y. The influence of α to the performance of the proposed funkcijų įvertinimų skaičiui (vidurkiui, medianai, didžiausiam) atliktam sustojimo kriterijui patenkinti; iš kiekvienos GKLS klasės buvo minimizuota po šimtą atsitiktinai sugeneruotų testinių funkcijų

#	Libre α	Vidurkis	Mediana	Didžiausia	#	Libre α	Vidurkis	Mediana	Didžiausia
1	0.01	185.94	140	810	4	0.01	2322.49	2042	6960
	0.2	**136.18**	**116**	428		0.2	**1142.39**	**949**	**3337**
	0.4	151.92	145	**371**		0.4	1448.94	1386	3484
	0.6	217.16	216	425		0.6	2976.24	2865	7554
	0.8	280.92	274	505		0.8	4788.87	4528	12745
	1.0	330.93	323	616		1.0	6796.57	6292	16722
2	0.01	940.08	909	2765	5	0.01	9994.97	6486	63849
	0.2	513.26	483	1459		0.2	**4793.12**	**3437**	25048
	0.4	**431.53**	**397**	1117		0.4	5339.45	4572	**16968**
	0.6	475.18	453	**935**		0.6	8033.69	7228	19282
	0.8	531.93	495	966		0.8	10415.30	9469	26604
	1.0	581.62	579	1060		1.0	12593.80	10595	33889
3	0.01	975.21	737	4738	6	0.01	27599.00	23961	98295
	0.2	**621.48**	**523**	2393		0.2	11188.10	9153	39736
	0.4	1009.82	957	**2113**		0.4	**8965.54**	**8422**	**23348**
	0.6	2071.96	2030	4360		0.6	10431.40	9606	25871
	0.8	3461.76	3191	7285		0.8	12421.80	10856	31172
	1.0	5021.94	4638	11517		1.0	14581.50	11764	38221

žymi uždavinio leistinosios srities dimensiją. Eksperimentai buvo atlikti naudojant 8 skirtingas testinių funkcijų klases. Klasės vadinamos „sunkiomis“, kai jų ρ yra santykinai didelis ir r yra santykinai mažas.

Pirmiausia buvo bandoma nustatyti tinkamą α parametro reikšmę. Eksperimentai buvo atliekami naudojant tokią pačią testavimo metodologiją kaip ir kitų autorių darbuose [27, 29, 37].

Minimizavimo proceas yra stabdomas po m-tosios tikslo funkcijos reikšmės įvertinimo, jeigu bet kurioje dimensijoje atstumas tarp globaliojo minimumo taško ir x_m taško, kuriame buvo apskaičiuota tikslo funkcijos reikšmė yra mažesnis nei iš anksto apibrėžtas dydis $\epsilon = \sqrt[d]{\delta}$. Algoritmas taip pat stabdomas, jeigu pasiekiamas 10^6 funkcijų įvertinimų skaičius ir prieš tai nurodytas sustojimo kriterijus nebūna patenkinamas. Visgi, mūsų eksperimentų metu šis sustojimo atvejis, kai globaliojo minimumo taškas prarandamas, nebuvo fiksuotas.

3 lentelė: Rezultatai gauti išsprendus 100 optimizavimo uždavinių iš kiekvienos GKLS testinių funkcijų klasės, naudojant sustojimo kriterijų, susietą su iš anksto žinomu globaliuoju minimumo tašku

Klasė	Sunkumas	Algoritmas	Vidurkis	Mediana	Didžiausia
1	Paprasta	DISIMPL-V	192.93	151.0	773
		GB-DISIMPL-V	174.25	151.0	472
		LIBRE α=0.4	**151.97**	**146.5**	**371**
2	Sunki	DISIMPL-V	1003.56	1021.0	2683
		GB-DISIMPL-V	518.11	**511.0**	1547
		LIBRE α=0.4	**431.55**	515.0	**1117**
3	Paprasta	DISIMPL-V	1061.83	787.0	4740
		GB-DISIMPL-V	**751.25**	**720.0**	2694
		LIBRE α=0.4	1009.72	959	**2113**
4	Sunki	DISIMPL-V	2598.91	2594.0	7354
		GB-DISIMPL-V	**1364.40**	**1283.0**	3723
		LIBRE α=0.4	1449.18	1390	**3484**
5	Paprasta	DISIMPL-V	10618.00	7334.0	58764
		GB-DISIMPL-V	**4579.24**	4615.0	**13825**
		LIBRE α=0.4	5340.43	**4575**	16968
6	Sunki	DISIMPL-V	33985.20	29807.0	118482
		GB-DISIMPL-V	10700.20	10033.0	30759
		LIBRE α=0.4	**8965.73**	**8436**	**23348**
7	Paprasta	DISIMPL-V	11200.4	7252	48590
		GB-DISIMPL-V	**5997.37**	**4524**	**23126**
		LIBRE α=0.4	17305.2	13343	65622
8	Sunki	DISIMPL-V	64751	42680	382593
		GB-DISIMPL-V	**28946**	**22416**	168067
		LIBRE α=0.4	44000.4	36306	**154277**

Įgyvendinti tokiam sustojimo kriterijui minimizuojamos tikslo funkcijos globaliojo minimimumo taškas turi būti žinomas; pažymėkime jį x^*; pastebėsime, kad $x^* \in D$. Sustojimo sąlyga yra patenkinama, kai taškas $x \in D$ yra rastas, kuris patenkina šią sąlygą:

$$|x_i - x_i^*| \leqslant \sqrt[d]{\delta}(u_i - l_i),\ i = 1, ..., d, \tag{6}$$

δ reikšmė yra pateikta 1 lentelėje.

α parametro reikšmės įtaka pasiūlytojo algoritmo efektyvumui buvo eksperimentiškai

įvertinta. Pasiūlytasis algoritmas buvo įvykdytas su skirtingomis $\alpha \in 0.01, 0.2, 0.4, 0.6, 0.8, 1.0$ reikšmėmis naudojant šešias atsitiktinai sugeneruotų GKLS testinių funkijų klases. Rezultatai yra pateikti 2 lentelėje.

Vertinant skirtingų testinių funkcijų klasių vidutinį ir didžiausią tikslo funkcijų įvertinimų skaičių matyti, kad reikšmė $\alpha = 0.4$ yra tinkama skirtingo sudėtingumo tikslo funkcijoms.

Pasiūlytojo algoritmo (kai $\alpha = 0.4$) veikimas palygintas su kitų algoritmų, kurie buvo kurti panašioms problemoms. Gautieji rezultatai pateikti 3 lentelėje kartu su DISIMPL-V ir GB-DISIMPL-V rezultatais paimtais iš straipsnių [27, 29]; geriausi rezultatai paryškinti.

Eksperimentų rezultatai rodo, kad pasiūlytasis algoritmas yra naudingas „sunkių“ klasių netinkamiausiais atvejais; žiūrėti stulpelį „Didžiausia“, pateiktą 3 lentelėje.

3.1.3 Modifikacijos

Šiame skyriuje pateikiamos pasiūlytojo algoritmo modifikacijos ir jos eksperimentiškai palyginamos. Pirmiausia, eksperimentiškai palyginami skirtingi surogatinių Lipšico rėžių apibrėžimai. Antra, palyginamos skirtingos strategijos parinkti posritis dalinimui. Galiausiai, eksperimentiškai palyginamos skirtingos strategijos apskaičiuoti Lipšico konstantos įvertį.

3.1.3.1 Lipšico rėžių įvertinimo strategijos

Išsikelta hipotezė, kad kuo griežtesni surogatiniai Lipšico rėžiai naudojami, tuo turėtų būti geresnis algoritmo efektyvumas. Kad patikrintumėme šią hipotezę pasiūlytojo algoritmo LIBRE kontekste, buvo pasirinktos keturios skirtingos strategijos apskaičiuoti surogatinius apatinius Lipšico rėžius po simpleksais.

Pasiūlytajame algoritme LIBRE apatiniai surogatiniai Lipšico rėžiai sudaromi naudo-

jant tik vieną simplekso viršūnę, kurioje yra mažiausia tikslo funkcijos reikšmė:

$$\widehat{L} = \alpha\widetilde{L}, \tag{7}$$

$$\boldsymbol{v}_{min}(S_i) = \arg\min_{\boldsymbol{v}_i \in V(S_i)} f(\boldsymbol{v}_i), \tag{8}$$

$$g(\boldsymbol{x}, S_i, \widehat{L}) = f(\boldsymbol{v}_{min}(S_i)) - \widehat{L}||\boldsymbol{x} - \boldsymbol{v}_{min}(S_i)||, \tag{9}$$

$$G(S_i, \widehat{L}) = \min_{\boldsymbol{x} \in S_i} g(\boldsymbol{x}, S_i, \widehat{L}) = f(\boldsymbol{v}_{min}(S_i)) - \widehat{L}\Delta(S_l). \tag{10}$$

Ši strategija buvo pasirinkta, nes ji buvo naudojama sėkminguose DISIMPL-V ir GB-DISIMPL-V [27, 29] algoritmuose. Ši strategija reikalauja mažai skaičiavimų, nes sprendinys 10) yra randamas analitiškai. 4 lentelėje ši strategija (8)-(10) yra pažymėta v_{min}.

Antroji pasirinktoji strategija (11)–(13) yra tokia pati kaip ir (8)–(10), išskyrus tai, kad viršūnė su didžiausia tikslo funkcijos reikšme yra naudojama sudaryti surogatiniams Lipšico rėžiams:

$$\boldsymbol{v}_{max}(S_i) = \arg\max_{\boldsymbol{v}_i \in V(S_i)} f(\boldsymbol{v}_i), \tag{11}$$

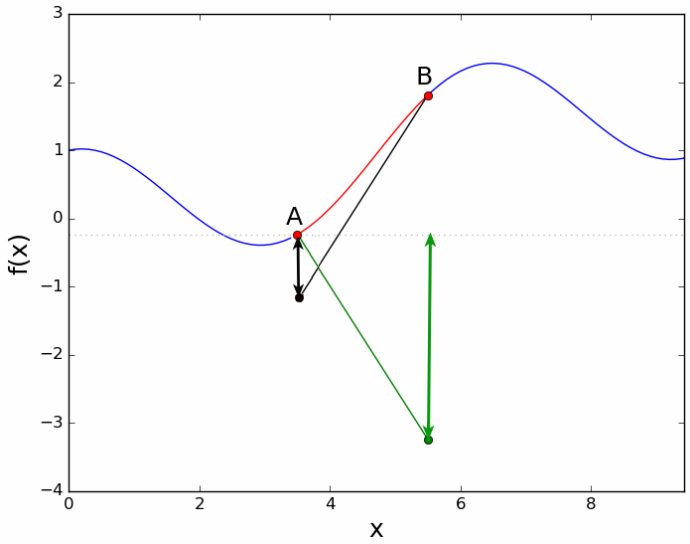
$$g(\boldsymbol{x}, S_i, \widehat{L}) = f(\boldsymbol{v}_{max}(S_i)) - \widehat{L}||\boldsymbol{x} - \boldsymbol{v}_{max}(S_i)||, \tag{12}$$

$$G(S_i, \widehat{L}) = \min_{\boldsymbol{x} \in S_i} g(\boldsymbol{x}, S_i, \widehat{L}) = f(\boldsymbol{v}_{max}(S_i)) - \widehat{L}\Delta(S_l). \tag{13}$$

4 lentelėje ši strategija (11)-(13) yra žymima v_{max}.

Šiuo atveju skaičiavimų apimtis išlieka tokia pati kaip ir pirmojoje strategijoje, tik yra išgaunami griežtesni surogatiniai Lipšico rėžiai. Pavyzdys vienmačiu atveju yra pateiktas 1 paveikslėlyje, kuriame žalia linija rodo surogatinius Lipšico rėžius, sudarytus naudojant strategiją v_{min}, ir juoda linija rodo surogatinius Lipšico rėžius, sudarytus naudojant strategiją v_{max}. Kaip galima matyti, antrosios strategijos rėžiai intervale yra griežtesni. Todėl maksimalus geriausios žinomos tikslo funkcijos reikšmės pagerinimas padalinus tą intervalą yra teisingai įvertinamas kaip mažesnis.

Trečioji strategija yra naudoti visas simplekso viršūnes surogatinių Lipšico rėžių su-



1 pav.: Vizualus strategijų v_{min} (žalioji linija) ir v_{max} (juoda linija) palyginamas vienmačiu atveju

darymui, nekreipiant dėmesio į papildomus skaičiavimus:

$$g^{strict}(\boldsymbol{x}, S_i, \widehat{L}) = \max_{\boldsymbol{v} \in V(S_i)} \{f(\boldsymbol{v}) - \widehat{L}||\boldsymbol{v} - \boldsymbol{x}||\}, \ \boldsymbol{x} \in S_i, \tag{14}$$

$$G^{strict}(S_i, \widehat{L}) = \min_{\boldsymbol{x} \in S_i} g^{strict}(\boldsymbol{x}, S_i, \widehat{L}). \tag{15}$$

Kad būtų randamas (15) sprendinys, vidinio lygmens optimizavimo uždavinys turi būti išsprendžiamas. Ši strategija užtikrina, kad būtų naudojami patys griežčiausi surogatiniai Lipšico rėžiai, kurie apskaičiuojami su aprėžto dydžio paklaida. Eksperimentuose buvo naudojama tolydaus padengimo strategija. Miesto kvartalų atstumas tarp padengiančiųjų taškų buvo pasirinktas ne mažesnis nei 10% simplekso diametro. Ši strategija žymima kaip *all_verts*.

Paskutinioji strategija yra bandymas rasti kompromisą tarp skaičiavimų apimties ir surogatinių Lipšico rėžių tikslumo. Pasiūlymas yra naudoti surogatinius Lipšico rėžius, apibrėžtus tik simplekso briaunose, kad būtų aproksimuojama (15) reikšmė simplekse:

$$G^{strict}(S_i, \widehat{L}) \approx \min_{\boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_i), i \neq j} G^{edge}(\boldsymbol{v}_i, \boldsymbol{v}_j, S_i, \widehat{L}) \ = \ \widetilde{G}^{strict}(S_i, \widehat{L}). \tag{16}$$

Surogatinių Lipšico rėžių briaunoje $(\boldsymbol{v}_i, \boldsymbol{v}_j)$ minimumas gali būti rastas analitiškai, jei-

4 lentelė: Skirtingų surogatinių Lipšico rėžių įvertinimo strategijų eksperimentinis palyginimas naudojant pirmąsias šešias su GKLS generatoriumi sugeneruotas testinių funkcijų klases

Klasė	Strategija	Vidurkis	Mediana	Didžiausia	Vidutinė įvertinimo trukmė sekundėmis
1	v_{min}	**151.92**	**145**	**371**	**0.000039**
	v_{max}	283.92	247	1002	0.000063
	all_verts	280.74	222	952	0.000349
	$edges$	204.58	162	512	0.000156
2	v_{min}	**431.53**	**397**	**1117**	**0.000071**
	v_{max}	1085.13	947	2757	0.00024
	all_verts	1297.6	1196	3133	0.000512
	$edges$	911.78	764	2361	0.000284
3	v_{min}	**1009.82**	**957**	**2113**	**0.000323**
	v_{max}	2205.68	1886	8156	0.001023
	all_verts	1942.33	1324	11803	0.004319
	$edges$	1371.71	989	4238	0.001714
4	v_{min}	**1448.94**	**1386**	**3484**	**0.000517**
	v_{max}	4771.85	4157	17699	0.002511
	all_verts	4605.42	3699	18843	0.007346
	$edges$	3133.99	2632	10821	0.004411
5	v_{min}	**5339.45**	**4572**	**16968**	**0.012432**
	v_{max}	11855.8	9490	52130	0.061572
	all_verts	11385	8784	61363	0.155959
	$edges$	10435.2	7701	58035	0.173710
6	v_{min}	**8965.54**	**8422**	**23348**	**0.035610**
	v_{max}	26246.3	16769	113266	0.156995
	all_verts	36930.1	28451	159262	0.582037
	$edges$	35082.5	28524	113353	0.522561

gu rėžiai sudaromi naudojant tik dvi viršūnes $\boldsymbol{v}_i$, $\boldsymbol{v}_j$ sudarančias briauną

$$G^{edge}(\boldsymbol{v}_i, \boldsymbol{v}_j, \widehat{L}) = \frac{f(\boldsymbol{v}_i) + f(\boldsymbol{v}_j) - \widehat{L}||\boldsymbol{v}_i - \boldsymbol{v}_j||}{2}, \tag{17}$$

$$X^{edge}(\boldsymbol{v}_i, \boldsymbol{v}_j, \widehat{L}) = \boldsymbol{v}_i t + \boldsymbol{v}_j(1-t), \;\; t = 0.5 + \frac{f(\boldsymbol{v}_j) - f(\boldsymbol{v}_i)}{2\widehat{L}||\boldsymbol{v}_j - \boldsymbol{v}_i||}, \tag{18}$$

čia $X^{edge}(\boldsymbol{v}_i, \boldsymbol{v}_j, \widehat{L})$ yra taškas, kuriame $G^{edge}(\boldsymbol{v}_i, \boldsymbol{v}_j, \widehat{L})$ reikšmė yra randama. Jeigu $G^{edge}(\boldsymbol{v}_i, \boldsymbol{v}_j, \widehat{L}) == g(X^{edge}(\boldsymbol{v}_i, \boldsymbol{v}_j, \widehat{L}), \widehat{L})$, tai $G^{edge}(\boldsymbol{v}_i, \boldsymbol{v}_j, S_i, \widehat{L}) = G^{edge}(\boldsymbol{v}_i, \boldsymbol{v}_j, \widehat{L})$. Kitu atveju, vienos dimensijos optimizavimo uždavinys turi būti sprendžiamas, kad būtų randama $G^{edge}(\boldsymbol{v}_i, \boldsymbol{v}_j, S_i, \widehat{L})$

aproksimacija. Mes naudojome Pijavskij-Shubert algoritmą [30, 39] su Lipšico konstanta, lygia $\widehat{L}$, ir laikėme problemą išspręsta, kai didžiausias tikėtinas reikšmės pagerėjimas buvo $\leqslant \epsilon = 10^{-4}$. 4 lentelėje ši strategija yra žymima kaip *edges*.

Veikimas su šiomis keturiomis strategijomis buvo eksperimentiškai įvertintas sprendžiant 600 testinių funkcijų iš skirtingų 6 testinių funkcijų klasių, sugeneruotų su GKLS generatoriumi [10] (žr. skyrių 3.1.2). Tas pats sustojimo kriterijus buvo naudojamas kaip ir [27, 37]. Veikimui įvertinti buvo apskaičiuotas vidutinis, mediana ir didžiausias įvertinimų skaičius, reikalingas patenkinti (6) sustojimo sąlygai. Skaičiavimų sudėtingumui įvertinti buvo naudojama vidutinė CPU trukmė vienam tikslo funkcijos įvertinimui atlikti. Skaitiniai rezultatai yra pateikti 4 lentelėje.

$\alpha = 0.4$ reikšmė buvo naudojama eksperimentuose (taip pat kaip ir 3.1.1 skyriuje). Eksperimentų rezultatai su skirtingomis α reikšmėmis yra pateikti 2 lentelėje.

Skaitiniai rezultatai (4 lentelėje) rodo, kad griežtesnių surogatinių apatinių Lipšico rėžių naudojimas nepadidino pasiūlytojo Libre algoritmo efektyvumo sprendžiant su GKLS generatoriumi sugeneruotų testinių uždavinių aibę. Surogatinių apatinių Lipšico rėžių, sudarytų tik iš viršūnės, kurioje yra geriausia tikslo funkcijos reikšmė, naudojimas yra geriausia strategija (euristika) tiek funkcijų įvertinimų skaičiaus, tiek ir skaičiavimų apimties atžvilgiu.

3.1.3.2 Potencialiai optimalių simpleksų parinkimas

Pasiūlytajame algoritme Libre yra naudojami du kriterijai (5) charakterizuoti ir parinkti potencialiai optimaliems simpleksams, kurie yra padalinami. Šiame skyriuje palyginamos dvi strategijos parinkti geriausiam kompromisui tarp šių dviejų kriterijų.

Pirmoji strategija yra parinkti visus Pareto optimalius sprendinius šių dviejų kriterijų (5) erdvėje

$$Y = \{(G(S_i), -\Delta(S_i)) : S_i \in S\}, \tag{19}$$

$$Y^{Pareto} = \left\{\boldsymbol{y} \in Y : \{\boldsymbol{y}' \in Y : \boldsymbol{y}' \succ \boldsymbol{y},\ \boldsymbol{y}' \neq \boldsymbol{y}\} = \varnothing\right\}, \tag{20}$$

$$P = \left\{S_i : S_i \in S, (G(S_i), -\Delta(S_i)) \in Y^{Pareto}\right\}, \tag{21}$$

5 lentelė: Strategijų potencialiai optimaliems simpleksams parinkti palyginimas naudojant pirmąsias šešias testinių funkcijų klases, sugeneruotas su GKLS testinių funkcijų generatoriumi

Klasė	Strategija	Vidurkis	Mediana	Didžiausia	Vidutinė įvertinimo trukmė sekundėmis
1	*Pareto*	153.42	149	421	**0.000039**
	Supported	**151.92**	**145**	**371**	**0.000039**
2	*Pareto*	464.42	408	1453	0.000086
	Supported	**431.53**	**397**	**1117**	**0.000071**
3	*Pareto*	**1004**	**953**	2200	0.000348
	Supported	1009.82	957	**2113**	**0.000323**
4	*Pareto*	1454.22	**1380**	3571	0.00055
	Supported	**1448.94**	1386	**3484**	**0.000517**
5	*Pareto*	5826.71	4994	**16870**	0.016318
	Supported	**5339.45**	**4572**	16968	**0.012432**
6	*Pareto*	9409.04	8947	26536	**0.029641**
	Supported	**8965.54**	**8422**	**23348**	0.035610

čia P yra aibė potencialiai optimalių simpleksų. Šią strategiją pažymėkime kaip *Pareto*.

Antroji strategija yra parinkti simpleksus, kurie atitinka iškilius Pareto optimalius sprendinius šių dviejų kriterijų (5) erdvėje

$$Y = \{(G(S_i), -\Delta(S_i)) : S_i \in S\}, \tag{22}$$

$$Y^{Supported} = Y \cap \{\boldsymbol{y} \in Conv(Y) : \{\boldsymbol{y}' \in Conv(Y) : \boldsymbol{y}' \succ \boldsymbol{y},\ \boldsymbol{y}' \neq \boldsymbol{y}\} = \varnothing\}, \tag{23}$$

$$P = \left\{S_i : S_i \in S, (G(S_i), -\Delta(S_i)) \in Y^{Supported}\right\}, \tag{24}$$

čia $Conv(Y)$ yra Y aibės iškilas apgaubimas (angl. convex hull). Ši strategija žymima kaip *Supported*.

Skaitiniai rezultatai (5 lentelėje) rodo, kad veikimas yra geresnis su sunkiais uždaviniais, kai yra naudojami iškilūs Pareto optimalūs sprendiniai, t. y. strategija *Supported*.

3.1.3.3 Lipšico konstantos įverčio globalumas

Jeigu visa aibė simpleksų, kuriais yra padengta leistinoji sritis, yra naudojama įvertinti Lipšico konstantai, tada gaunamas *globalus Lipšico konstantos įvertis*. Kitu atveju, kai

padengimo poaibis yra naudojamas, tada gaunamas *lokalus Lipšico konstantos įvertis*. Kai Lipšico konstanta yra įvertinama lokaliai, ji gali reprezentuoti lokalią sritį tiksliau nei globalus Lipšico konstantos įvertis. Pavyzdžiui, jeigu funkcijos reikšmės srityje būtų seklios, tada lokalus Lipšico konstantos įvertis būtų mažesnis. Taigi, tai turėtų sumažinti šios srities tyrinėjimo prioritetą, ir taip galėtų sumažėti nereikalingų tikslo funkcijos įvertinimų skaičius. Šiame skyriuje yra išsikeliama hipotezė, kad lokalios Lipšico konstantos naudojimas turėtų pagerinti pasiūlytojo algoritmo Libre efektyvumą. Norint patikrinti šią hipotezę buvo pasiūlytos dvi papildomos strategijos, kaip galima įvertinti ir naudoti Lipšico konstantos įvertį.

Pasiūlytame Libre algoritme naudojamas tik globalus Lipšico konstantos įvertis, kuris yra apibrėtas taip

$$\widetilde{L}_{global} = \max\left\{\frac{|f(\boldsymbol{v}_i) - f(\boldsymbol{v}_j)|}{||\boldsymbol{v}_i - \boldsymbol{v}_j||} : \boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_l), \boldsymbol{v}_i \neq \boldsymbol{v}_j, S_l \in S\right\} \tag{25}$$

čia S yra simpleksų aibė padengianti leistinąją sritį ir $V(S_l)$ yra S_l viršūnių aibė. 6 lentelėje ši strategija yra žymima kaip *global*.

Antroji strategija yra naudoti tik lokalųjį Lipšico konstantos įvertį. Kai surogatiniai Lipšico rėžiai yra skaičiuojami tam tikram simpleksui S_i, tik jo lokalus Lipšico konstantos įvertis yra naudojamas

$$\widetilde{L}_{local}(S_i) = \max\left\{\frac{|f(\boldsymbol{v}_k) - f(\boldsymbol{v}_l)|}{||\boldsymbol{v}_k - \boldsymbol{v}_l||} : \boldsymbol{v}_k, \boldsymbol{v}_l \in V(S_j),\ V(S_j) \cap V(S_i) \neq \varnothing,\ S_j \in S\right\}. \tag{26}$$

Sritis, kurioje lokalus Lipšico konstantos įvertis yra įvertinamas, susideda iš simplekso ir jo kaimyninių simpleksų. Šiuo atveju simpleksai laikomi kaimyniniais, jeigu jie turi bent vieną bendrą viršūnę. 6 lentelėje ši strategija žymima *local*.

Trečioji strategija yra kombinuoti globaliųjį ir lokalųjį Lipšico konstantų įverčius. Buvo pasirinkta ta pati kombinavimo strategija kaip ir informacijos globalaus optimizavimo algoritme su lokaliu sureguliavimu [32]:

$$\widetilde{L}_{mixed}(S_i) = \max\{\widetilde{L}_{local},\ \widetilde{L}_{global}\frac{\Delta(S_i)}{\Delta_{max}}\}, \tag{27}$$

$$\Delta_{max} = \max_{S_j \in S} \Delta(S_j). \tag{28}$$

6 lentelė: Skirtingų Lipšico konstantos įverčio radimo strategijų palyginimas naudojant pirmąsias šešias testinių funkcijų klases sugeneruotas naudojant GKLS generatorių

Klasė	Strategija	Vidurkis	Mediana	Didžiausia	Vidutinė įvertinimo trukmė sekundėmis
1	*global*	151.92	145	371	**0.000039**
	local	**122.04**	**116**	265	0.000049
	mixed	126.87	121	**257**	0.000047
2	*global*	431.53	397	1117	**0.000071**
	local	**290.16**	**261**	**888**	0.000086
	mixed	292.25	269	1011	0.000089
3	*global*	**1009.82**	**957**	**2113**	**0.000323**
	local	1153.69	1087	2587	0.000693
	mixed	1093.65	1066	2304	0.000713
4	*global*	1448.94	1386	3484	**0.000517**
	local	**1181.18**	**1158**	2439	0.000753
	mixed	1186.78	1156	**2163**	0.000834
5	*global*	**5339.45**	**4572**	**16968**	**0.012432**
	local	12715.1	12599	31046	0.083735
	mixed	11023.1	10797	24078	0.075724
6	*global*	**8965.54**	**8422**	**23348**	**0.035610**
	local	11594.5	11198	27905	0.078067
	mixed	10989.4	10771	23630	0.066359

Šiuo atveju naudojamo Lipšico konstantos įverčio globalumas yra susiejamas su santykiniu simplekso dydžiu. $\widetilde{L}_{global}$ visada yra naudojamas didžiausiems padengimo simpleksams (nes jų santykinis dydis visada yra 1 ir $\widetilde{L}_{local} \leqslant \widetilde{L}_{global}$). Mažesniems simpleksams yra naudojama reikšmė $\geqslant \widetilde{L}_{local}$ ir $< \widetilde{L}_{global}$. 6 lentelėje ši strategija žymima *mixed*.

Šioms strategijoms palyginti buvo naudojama tokia pati eksperimentinė metadologija, kaip ir ankstesniuose skyriuose. T. y. buvo sprendžiamos 600 su GKLS generatoriumi atsitiktinai sugeneruotų testinių uždavinių; fiksuojamas vidutinis, mediana ir didžiausias tikslo funkcijos įvertinimų skaičius bei vidutinė CPU trukmė, reikalinga vienam tikslo funkcijos įvertinimui atlikti. Skaitiniai rezultatai pateikti 6 lentelėje. $\alpha = 0.4$ reikšmė buvo naudojama šiuose eksperimentuose.

Iš skaitinių rezultatų, pateiktų 6 lentelėje, galima matyti, kad nė viena iš šių trijų strategijų neišsiskyrė kaip aiškiai efektyvesnė visoms testinių uždavinių klasėms. Visgi,

efektyvumas su strategija *global* buvo geresnis tiek 5-tai, tiek 6-tai klasėms, kurios turi didžiausią dimensijų skaičių iš nagrinėtų uždavinių klasių. Taip pat galima pastebėti, kad lokalios Lipšico konstantos įverčio naudojimas padidino skaičiavimų apimtį, nes vidutinė trukmė, tenkanti tikslo funkcijos įvertinimui, padidėjo.

3.2 Globaliojo optimizavimo algoritmas, naudojantis lokalųjį Lipšico konstantos įvertį

Nepriklausomai nuo Libre algoritmo, buvo pasiūlytas kitas panašus algoritmas [12], kuriame naudojamas tik lokalus Lipšico konstantos įvertis, vietoje globalaus. Pagrindiniai šio algoritmo ir Libre algoritmo modifikacijos *local*, aprašytos 3.1.3.3 skyriuje, skirtumai yra:

1. Naudojamas Lipšico konstantos įvertis yra maksimumas šių dydžių: kaimyninių simpleksų viršūnių poroms apskaičiuoto dydžio (3) bei mažiausią tikslo funkcijos reikšmę turinčioje viršūnėje apskaičiuotos simpleksinio gradiento normos L', kuri simplekso S_k viršūnėje v_1 gali būti apskaičiuojama taip:

$$B(S_k) = (\boldsymbol{v}_2 - \boldsymbol{v}_1, \boldsymbol{v}_3 - \boldsymbol{v}_1, ..., \boldsymbol{v}_{d+1} - \boldsymbol{v}_1), \tag{29}$$

$$F(S_k, f) = (f(\boldsymbol{v}_2) - f(\boldsymbol{v}_1), f(\boldsymbol{v}_3) - f(\boldsymbol{v}_1), ..., f(\boldsymbol{v}_{d+1}) - f(\boldsymbol{v}_1))^T, \tag{30}$$

$$\boldsymbol{v}_i \in V(S_k),\ i = 1, ..., d+1, \tag{31}$$

$$L' = ||B(S_l)^{-T} F(S_l, f)||. \tag{32}$$

2. Surogatiniai Lipšico rėžiai sudaromi naudojant visas simplekso viršūnes:

$$g(\boldsymbol{x}) = \max_{\boldsymbol{v} \in V(S)} \{f(\boldsymbol{v}) - L||\boldsymbol{v} - \boldsymbol{x}||, \boldsymbol{x} \in S\} \tag{33}$$

   (taip pat kaip ir (14)). Šių rėžių minimumui rasti sprendžiamas vidinis optimizavimo uždavinys.

3. Simpleksai laikomi kaimyniniais, jeigu skiriasi ne daugiau kaip dvi jų viršūnės.

4. Algoritme nenaudojamas α parametras, kuris keistų Lipšico konstantos įverčio reikšmę.

Algorithm 2: Pasiūlytojo globaliojo optimizavimo algoritmo, naudojančio lokalųjį Lipšico konstantos įvertį, aprašymas

Padengti leistinąją sritį D nepersidengiančiais simpleksais $S = \{S_i : D = \cup S_i,\ i = 1, ..., d!\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.
Įvertinti $\{f(v_i) : v_i$ unikalios S viršūnės, $i = 1, ..., 2^d\}$.
while *sustojimo sąlyga nėra patenkinta* **do**
    foreach $S_l \in S$ **do**
        Rasti $\widehat{L}_l = \max\left\{\frac{|f(v_i)-f(v_j)|}{||v_i-v_j||} : v_i, v_j \in V(S_l), v_i \neq v_j\right\}$.
    foreach $S_l \in S$ **do**
        Rasti $B(S_l) = (v_2 - v_1, v_3 - v_1, ..., v_{d+1} - v_1),\ v_1 = v_{min},\ v_i \in V(S_l)$,
        $F(S_l, f) = (f(v_2) - f(v_1), f(v_3) - f(v_1), ..., f(v_{d+1}) - f(v_1))^T$,
        $\widetilde{L}_l =$
        $\max\left\{\widehat{L}_j : S_j, S_l \textit{ turi } \leqslant 2 \textit{ skirtingas virsunes}\right\} \cup \left\{||B(S_l)^{-T}F(S_l, f)||\right\}$.
        Rasti $G_l = G(S_l, \widetilde{L})$, sprendžiant vidinį optimizavimo uždavinį.
    Parinkti aibę simpleksų dalinimui: $P = \{S_i : S_i \in S,\ S_i$ priklauso
      $\min(G_l, -\Delta(S_l))$ iskiliesiems Pareto optimaliems sprendiniams$\}$
    foreach $S_l \in P$ **do**
        Padalinti S_l į du naujus simpleksus S_l^1, S_l^2, pridedant viršūnę v ilgiausios S_l briaunos vidurio taške.
        Atnaujinti $S = S\backslash\left\{S_l\right\} \cup \left\{S_l^1, S_l^2\right\}$. Jei v nauja viršūnė, įvertinti $f(v)$.
        Atnaujinti f_{min}.
return f_{min}

Algoritmo pseudokodas pateiktas 2 algoritme. Kadangi vidinio optimizavimo uždavinio leistinoji sritis yra simplekso formos ir funkcijų įvertinimai yra santykinai pigūs, buvo pasirinkta šį uždavinį spręsti algoritmu Vidinis, pateiktu 3 algoritme.

Metodo efektyvumui ištirti buvo pasirinkta tokia pati eksperimentinė metodologija kaip ir aprašyta 3.1.2 skyriuje. Rezultatai su pirmomis keturiomis GKLS testinėmis funkcijų klasėmis pateiktos 7 lentelėje. Iš šios lentelės galima matyti, kad šiame skyriuje aprašyto algoritmo rezultatai yra geresni nei Libre algoritmo vidutiniu funkcijų įvertinimų skaičiaus ir medianos atžvilgiu su pirmomis keturiomis GKLS testinių funkcijų klasėmis. Visgi didesnės dimensijos problemoms su šiuo algoritmu gaunami rezultatai yra prastesni nei Libre. Galima daryti išvadą, kad algoritmų, kuriuose naudojamas lokalusis Lipšico konstantos įvertis, efektyvumui poveikį daro kaimyninių simpleksų apibrėžimo pasirinkimas, todėl reikėtų išbandyti daugiau skirtingų

Algorithm 3: Vidinio lygmens optimizavimo algoritmas

Input S_k – simpleksas, kurio rėžių reikšmė minimizuojama, $\widetilde{L}$ – Lipšico konstantos įvertis, g – surogatinių Lipšico rėžių funkcija, M_{max} – maksimalus iteracijų skaičius (numatytoji reikšmė 10).
Function *VIDINIS*(S_k, $\widetilde{L}$, g $M_{max} = 10$)
  $S = \{S_k\}$
  $g_{min} = \min\{g(\boldsymbol{v}) : \boldsymbol{v} \in V(S_k)\}$
  $m = 1$
  while $m < M_{max}$ **do**
    foreach $S_l \in S$ **do**
      Rasti $G(S_l) = max\{g(l_1) - L\Delta(S_l), g(l_2) - L\Delta(S_l)\}$, čia l_1, l_2 - ilgiausios S kraštinės viršūnės.
    Parinkti simpleksą dalinimui: $S_p = \arg\min_{S_l \in S} G(S_l)$.
    Padalinti S_p į du naujus simpleksus S_p^1, S_p^2, pridedant viršūnę $\boldsymbol{v}$ ilgiausios S_p briaunos vidurio taške.
    Atnaujinti $S = S \backslash \{S_p\} \cup \left\{S_p^1, S_p^2\right\}$. Jei $\boldsymbol{v}$ nauja viršūnė, įvertinti $g(\boldsymbol{v})$ ir atnaujinti g_{min}.
    $m = m + 1$
  return g_{min}.

kaimyninių simpleksų apibrėžimų.

3.3 Strategija vieno-kriterijaus Lipšico optimizavimo algoritmams apibendrinti daugiakriteriniam atvejui

Šiame skyriuje yra aprašoma bendra strategija apibendrinti vienkriterinius Lipšico globaliojo optimizavimo algoritmus daugiakriteriniam atvejui. Ši strategija yra pritaikoma Lipšico globaliojo optimizavimo algoritmams, kurie iteratyviai atnaujina leistinosios srities D padengimą S:

$$S = \{S_i : D = \cup S_i, i = 1, ..., k\}, \tag{34}$$

taip, kad kiekvienos iteracijos metu vienas ar daugiau posričių yra parenkami dalinimui remiantis minimalia apatinių Lipšico rėžių reikšme posrityje (žr. 2 paveikslą, kuriame pateikta bendra tokio algoritmo schema). Kuo žemesni Lipšico rėžiai, tuo mažesnę (geresnę) funkcijos reikšmę yra įmanoma gauti padalinus tą posritį. Šiame darbe

7 lentelė: Rezultatai gauti išsprendus 100 optimizavimo uždavinių iš keturių pirmųjų GKLS testinių funkcijų klasių, naudojant sustojimo kriterijų, susietą su iš anksto žinomu globaliuoju minimumo tašku

Klasė	Sunkumas	Algoritmas	Vidurkis	Mediana	Didžiausia
1	Paprasta	DISIMPL-V	192.93	151.0	773
		GB-DISIMPL-V	174.25	151.0	472
		LIBRE α=0.4	151.97	146.5	371
		Pasiūlytasis	**103.68**	**91**	**277**
2	Sunki	DISIMPL-V	1003.56	1021.0	2683
		GB-DISIMPL-V	518.11	511.0	1547
		LIBRE α=0.4	431.55	515.0	**1117**
		Pasiūlytasis	**384.02**	**298.5**	1714
3	Paprasta	DISIMPL-V	1061.83	787.0	4740
		GB-DISIMPL-V	**751.25**	**720.0**	2694
		LIBRE α=0.4	1009.72	959	2113
		Pasiūlytasis	963.21	866.5	**2162**
4	Sunki	DISIMPL-V	2598.91	2594.0	7354
		GB-DISIMPL-V	1364.40	1283.0	3723
		LIBRE α=0.4	1449.18	1390	3484
		Pasiūlytasis	**1079.56**	**1029.5**	**2859**

yra naudojami tokie žymėjimai apatiniams Lipšico rėžiams ir jų minimumo reikšmei tam tikrame posrityje S_i:

$$g(\boldsymbol{x}, S_i),\ \boldsymbol{x} \in S_i, \tag{35}$$

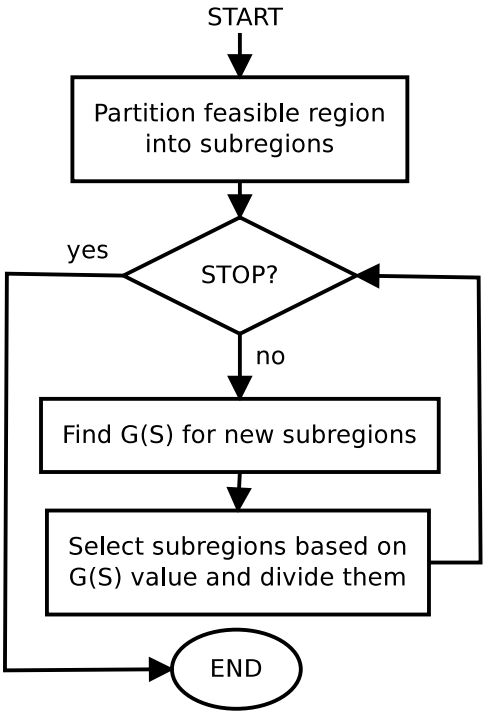
$$G(S_i) = \min_{\boldsymbol{x} \in S_i} g(\boldsymbol{x}, S_i). \tag{36}$$

Pavyzdžiui, $g(\boldsymbol{x}, S_i)$ ir $G(S_i)$ reikšmės naudojamos straipsniuose [27, 29] ir LIBRE algoritme yra:

$$g^{single}(\mathbf{x}, S_i) = \min_{\mathbf{v} \in V(S_i)} f(\mathbf{v}) - L||\mathbf{x} - \arg\min_{\mathbf{v} \in V(S_i)} f(\mathbf{v})||,\ \mathbf{x} \in S_i. \tag{37}$$

$$G^{single}(S_i) \approx \min_{\mathbf{v} \in V(S_i)} f(\mathbf{v}) - L\Delta(S_i) \quad \leqslant \quad \min_{\mathbf{x} \in S_i} g^{single}(\mathbf{x}, S_i). \tag{38}$$

didžiausias tikėtinas turimo sprendinio f_{min} pagerėjimas padalinus S_i yra lygus $f_{min} - G(S_i)$. Didžiausias tikėtinas turimo sprendinio pagerėjimas turi būti maksimizuoja-



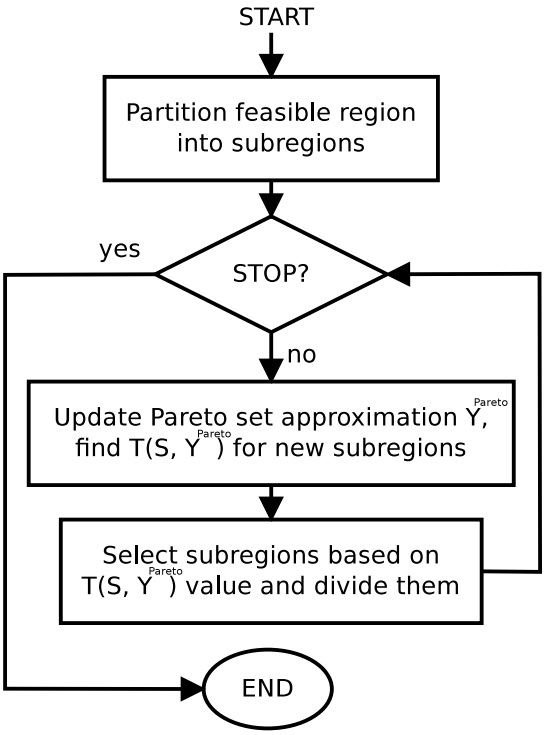
2 pav.: Bendra Lipšico globaliojo optimizavimo algoritmo schema

mas, o jeigu norima minimizuoti, gaunamas kriterijus $-(f_{min} - G(S_i)) = G(S_i) - f_{min}$. Norint tarpusavyje palyginti posritis pakanka palyginti $-G(S_i)$ ar $G(S_i)$ posričių reikšmes, nes f_{min} reikšmė yra fiksuota kiekvienoje iteracijoje.

Daugiakriteriniu atveju kriterijus, apibrėžiantis posričių tinkamumą tolimesniam dalinimui, nėra trivialus, nes keletas prieštaringų tikslo funkcijų turi būti apsvarstoma vienu metu. Vektorius sudarytas iš apatinių Lipšico rėžių kiekvienai tikslo funkcijai:

$$\mathbf{g}(\boldsymbol{x}, S_i) = (g^1(\boldsymbol{x}, S_i), ..., g^n(\boldsymbol{x}, S_i)). \tag{39}$$

Buvo bandymų apibrėžti posričių parinkimo dalinimui kriterijų kaip daugiakriterinių Lipšico rėžių griežtumą [42, 43]. Šiuose darbuose buvo pasiūlytas įvertis, kuris apibrėžia, kokie griežti Lipšico rėžiai posrityje yra: kuo įvertis mažesnis, tuo rėžiai griežtesni. Norint visoje apibrėžimo srityje griežtinti Lipšico rėžius, reikia dalinimui parinkti posritį, kuris turi didžiausią griežtumo įverčio reikšmę. Vienmačiu atveju, kai S_i yra atkarpa ir $V(S_i)$ yra tos atkarpos kraštai, Lipšico rėžių griežtumo įvertis,



3 pav.: Bendra daugiakriterinio Lipšico optimizavimo algoritmo, kuris gaunamas pritaikius pasiūlytąją strategiją apibendrinti vienkriterinio Lipšico algoritmą daugiakriteriniam atvejui, schema

nauojant mūsų pasižymėjimą, gali būti išreikštas taip:

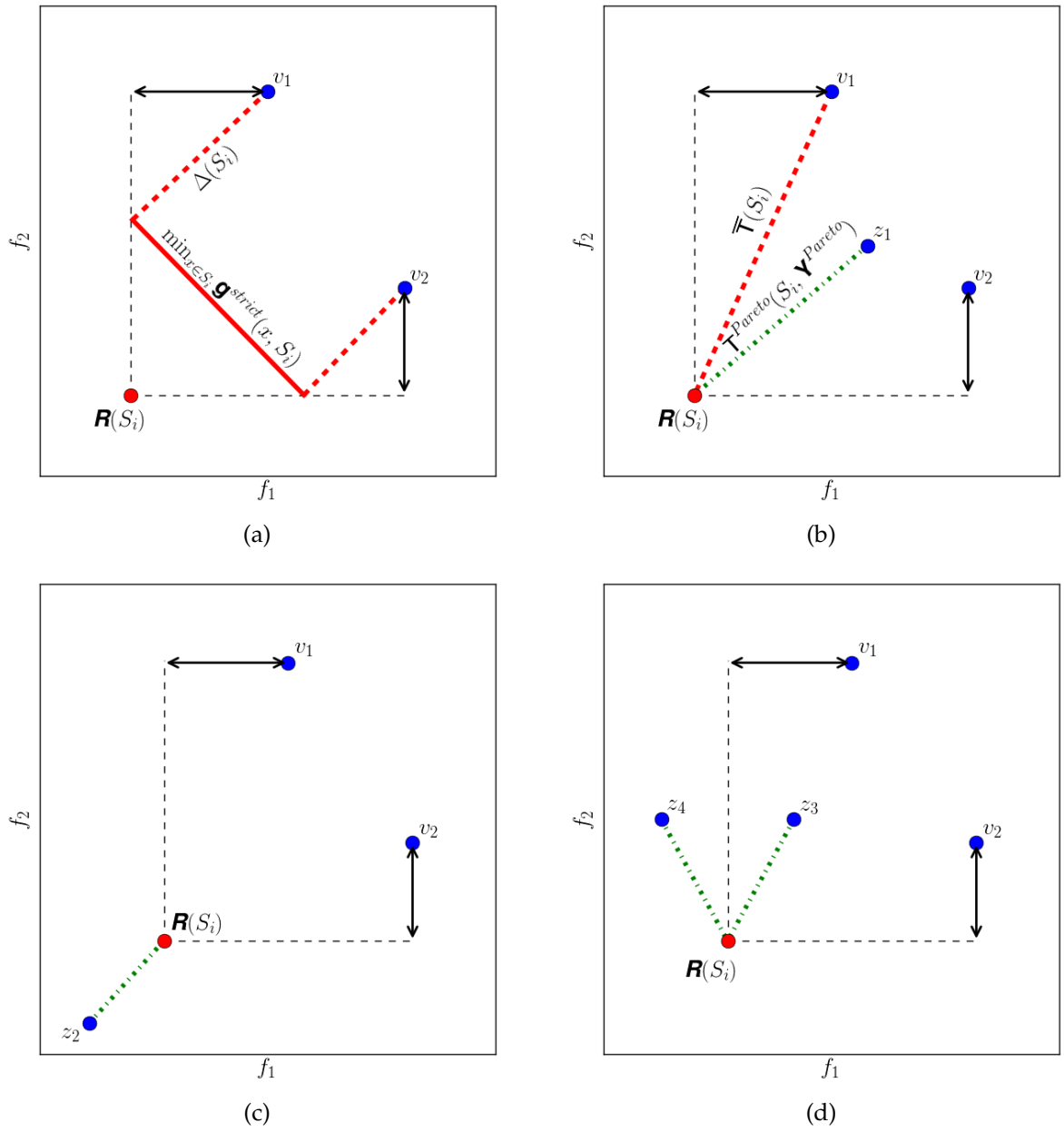
$$T(S_i) = \max\left\{ \min_{\xi \in \boldsymbol{G}^{strict}(S_i)} ||\xi - \boldsymbol{f}(\boldsymbol{v})|| : \boldsymbol{v} \in V(S_i) \right\}, \tag{40}$$

$$\boldsymbol{G}^{strict}(S_i) = \min_{\boldsymbol{x} \in S_i} \boldsymbol{g}^{strict}(\boldsymbol{x}, S_i), \tag{41}$$

$$\boldsymbol{g}^{strict}(\boldsymbol{x}, S_i) = (g^{strict,1}(\boldsymbol{x}, S_i), \ldots, g^{strict,n}(\boldsymbol{x}, S_i)), \tag{42}$$

$$g^{strict}(\boldsymbol{x}, S_i) = \max_{\boldsymbol{v} \in V(S_i)} \{f(\boldsymbol{v}) - L||\boldsymbol{v} - \boldsymbol{x}||\}, \ \boldsymbol{x} \in S_i, \tag{43}$$

čia $\|\cdot\|$ yra Euklidinė norma. Šio apibrėžimo iliustracija vieno kintamojo dvikriteriniam atvejui yra pateikta 4a paveiksle, kuriame $V(S_i) = \{\boldsymbol{v}_1, \boldsymbol{v}_2\}$. Šiuo atveju taškų aibė tikslo funkcijų erdvėje (41) yra atkarpa. Lipšico rėžių griežtumas gali būti suprantamas kaip didžiausias atstumas tarp atkarpos ir taškų $V(S_i)$ tikslo funkcijų erdvėje. Šis apibrėžimas yra paprastas, kai tikslo funkcijų yra dvi, tačiau kai tikslo funkcijų yra daugiau, aibė (41) yra hiperpaviršius ir jo sudėtingumas sukelia problemų naudoti



4 pav.: Vienmačio posričio apatiniai Lipšico rėžiai dviejų tikslo funkcijų erdvėje. v_1, v_2 yra S_i viršūnės.

kriterijų (40).

Kad būtų galima išvengti šio apribojimo, gali būti naudojamas vienas taškas apriboti posričio apatiniams Lipšico rėžiams. Konkrečiau, kiekvienai tikslo funkcijai atskirai galima rasti jos Lipšico rėžių posrityje minimumo tašką, taigi apribojančio taško api-

brėžimas:

$$\boldsymbol{R}(S_i) = (G^1(S_i), ..., G^n(S_i)) = (\min g^1(S_i), ..., \min g^n(S_i)). \tag{44}$$

Lipšico rėžių griežtumo įverčio apibrėžimas gali būti pritaikytas naudoti):

$$\overline{T}(S_i) = \max_{\boldsymbol{v} \in V(S_i)} ||\boldsymbol{f}(\boldsymbol{v}) - \boldsymbol{R}(S_i)||. \tag{45}$$

4b paveiksle pavaizduotas kitas vienmačio dviejų kriterijų uždavinio atvejis, kai diskrečią Pareto fronto aproksimacija yra $Y^{Pareto} = \{\mathbf{f}(\mathbf{v}_1), \mathbf{f}(\mathbf{v}_2), \mathbf{z}_1\}$ ir $V(S_i) = \{\mathbf{v}_1, \mathbf{v}_2\}$. Lipšico rėžių griežtumo įvertis $\overline{T}(S_i)$ yra lygus didžiausiam atstumui tarp apribojančiojo taško $\boldsymbol{R}(S_i)$ ir tikslo funkcijų reikšmių viršūnėse $V(S_i)$.

Apibrėžimai (40) ir (45) atspindi didžiausią tikėtiną tikslo funkcijų reikšmių viršūnėse $V(S_i)$ pagerėjimą, kuris įmanomas padalinus posritį S_i. Tačiau, šis apibrėžimas neįvertina kitų Y^{Pareto} taškų, tarp kurių gali būti taškų, kurių atžvilgiu tikėtinas pagerėjimas yra kur kas mažesnis. Pavyzdžiui, 4b paveiksle galima matyti, kad $\overline{T}(S_i) < T^{Pareto}(S_i, Y^{Pareto})$. Šiai problemai spręsti siūlome Lipšico rėžių griežtumo įvertį išreikšti kaip galimą didžiausią Y^{Pareto} aibės pagerėjimą:

$$T^{Pareto}(S_i, Y^{Pareto}) = \min_{\boldsymbol{y} \in Y^{Pareto}} ||\boldsymbol{y} - \boldsymbol{R}(S_i)||. \tag{46}$$

Šioje formulėje minimizavimas atspindi faktą, kad S_i padalinimas gali pagerinti tašką iš Pareto fronto aproksimacijos, kuris yra arčiausiai $\boldsymbol{R}(S_i)$.

4b paveiksle pavaizduotas apibrėžimo (46) pavyzdys, kuriame Lipšico rėžių griežtumas $T^{Pareto}(S_i, Y^{Pareto})$ yra lygus atstumui tarp apribojančiojo taško $\boldsymbol{R}(S_i)$ ir $\boldsymbol{z}_1 \in Y^{Pareto}$.

Toliau siūlome dar du (46) apibrėžimo patobulinimus. Pirmiausia apsvarstykime atvejį, kai egzistuoja taškas $\boldsymbol{x}_2 \in Y^{Pareto}$, kuris dominuoja $\boldsymbol{R}(S_i)$ (žr. 4c paveikslą). Šiuo atveju išraiška (46) yra teigiama, tai turėtų reikšti, kad dalinant S_i įmanoma pagerinti Pareto aibės tašką, kas nėra tiesa. Į atnaujintą (46) apibrėžimą įtraukiamos neigiamos reikšmės, kuriomis nurodoma, kad $\boldsymbol{R}(S_i)$ yra dominuojamas:

$$T'(S_i, Y^{Pareto}) = \begin{cases} \min\limits_{\boldsymbol{y} \in Y^{Pareto}} \|\boldsymbol{y} - \boldsymbol{G}(S_i)\|, & \text{if } \nexists \boldsymbol{y}' \succ \boldsymbol{G}(S_i),\ \boldsymbol{y}' \in Y^{Pareto}, \\ -\min\limits_{\boldsymbol{y} \in Y^{Pareto}} \|\boldsymbol{y} - \boldsymbol{G}(S_i)\|, & \text{if } \exists \boldsymbol{y}' \succ \boldsymbol{G}(S_i),\ \boldsymbol{y}' \in Y^{Pareto}. \end{cases} \tag{47}$$

Antra, apsvarstykime atvejį, kai du skirtingi taškai $z_3, z_4 \in Y^{Pareto}$ yra vienodai nutolę nuo $\boldsymbol{R}(S_i)$ (žr. 4d paveikslą). Reikšmė (47) posričiai S_i yra tokia pati tiek z_3, tiek z_4. Tačiau, $\boldsymbol{R}(S_i)$ yra geresnis už z_3 visų tikslo funkcijų atžvilgiu, , tuo tarpu už z_4 geresnis tik vienos tikslo funkcijos reikšmės atžvilgiu. Šis pavyzdys parodo, kad (47) apibrėžimas turėtų būti atnaujintas, kad būtų atsižvelgiama tik tikslo funkcijas, kurių reikšmėmis $\boldsymbol{R}(S_i)$ yra geresnis. Be to, jeigu $\boldsymbol{R}(S_i)$ yra geresnis bent vienos tikslo funkcijos atžvilgiu, kitų tikslo funkcijų reikšmių pabloginimai neturėtų turėti įtakos. Siekiamas efektas gali būti išgautas pakeičiant Euklidinę normą (47) formulėje asimetrine norma $|||\cdot|||$, kuri ignoruotų reikšmių pabloginimą. Siūlome kelis asimetrinės normos variantus:

$$|||\boldsymbol{a}-\boldsymbol{b}|||_1 = \sum_{j=1}^{k} p(a_j, b_j), \quad \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^k, \tag{48}$$

$$|||\boldsymbol{a}-\boldsymbol{b}|||_2 = \sqrt{\sum_{j=1}^{k} p(a_j, b_j)^2}, \quad \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^k, \tag{49}$$

$$|||\boldsymbol{a}-\boldsymbol{b}|||_\infty = \max_{j=1,\dots,k} p(a_j, b_j), \quad \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^k, \tag{50}$$

$$p(a, b) = \max(0,\, a-b), \quad a, b \in \mathbb{R}. \tag{51}$$

Atnaujintas (47) apibrėžimas:

$$T(S_i, Y^{Pareto}) = \begin{cases} \min\limits_{\boldsymbol{y} \in Y^{Pareto}} |||\boldsymbol{y} - \boldsymbol{R}(S_i)|||_2, & \text{if } \nexists \boldsymbol{y}' \succ \boldsymbol{R}(S_i),\ \boldsymbol{y}' \in Y^{Pareto}, \\ -\min\limits_{\boldsymbol{y} \in Y^{Pareto}} |||\boldsymbol{R}(S_i) - \boldsymbol{y}|||_2, & \text{if } \exists \boldsymbol{y}' \succ \boldsymbol{R}(S_i),\ \boldsymbol{y}' \in Y^{Pareto}, \end{cases} \tag{52}$$

kuris gali būti interpretuojamas kaip mažiausias atstumas tarp $\boldsymbol{R}(S_i)$ ir Y^{Pareto}.

Norint apibendrinti Lipšico globaliojo optimizavimo algoritmą daugiakriteriniam atvejui, kiekvienos algoritmo iteracijos metu turi būti atnaujinama Pareto aibės aproksimacija Y^{Pareto}. Be to, tolimesniam dalinimui posričiai turėtų būti parenkami naudojant $T(S_i, Y^{Pareto})$ (52) kriterijų, vietoje $G(S_i)$, taip įvertinant didžiausią tikėtiną dabartinio sprendinio pagerinimą, kuris įmanomas padalinant S_i. Galima pastebėti, kad pritaikius šiuos algoritmo pakeitimus, vieno kriterijaus uždaviniams algoritmo rezultatai išlieka tokie patys kaip ir prieš pritaikant pakeitimus. Taip yra, nes vieno kriterijaus atveju $Y^{Pareto} = \{f_{min}\}$ ir $T(S_i, Y^{Pareto}) = f_{min} - G(S_i)$. $T(S_i, Y^{Pareto})$ turi būti maksimi-

zuojamas, o jeigu norima minimizuoti, gaunama $-T(S_i, Y^{Pareto}) = G(S_i) - f_{min}$, t. y. gaunamas tas pats kriterijus, kaip ir prieš pritaikant apibendrinimo strategiją. Kaip jau buvo minėta, f_{min} yra konstanta kiekvienos algoritmo iteracijos metu, tai $G(S_i) - f_{min}$ atitinka $G(S_i)$ reikšmė, kuri yra pakeista per konstantą, o tai nedaro įtakos palyginimo rezultatui.

3.4 Daugiakriterinis Libre algoritmo atvejis

3.4.1 Vienkriterinio ir daugiakriterinio Libre versijų skirtumai

Strategija (aprašyta ankstesniame 3.3 skyriuje), skirta apibendrinti vieno kriterijaus Lipšico optimizavimo algoritmams daugiakriteriniam atvejui, buvo pritaikyta pasiūlytajam Libre algoritmui. Apibendrinus algoritmą daugiakriteriniam atvejui, jo veikimas vieno kriterijaus atveju nepasikeitė, tačiau algoritmas įgavo galimybę spręsti ir daugiakriterinius uždavinius. Turėdami tai omenyje, ir daugiakriteriniam algoritmo atvejui paliksime tą patį Libre pavadinimą. Gautojo daugiakriterinio algoritmo pseudokodas yra pateiktas 4 algoritme. Šis algoritmas buvo realizuotas C++ programavimo kalba ir išeities kodas su AGPL licencija yra pateiktas kaip disertacijos priedas. Likusi skyriaus dalis yra skirta akcentuoti kai kuriuos daugiakriterinio algoritmo versijos aspektus.

Lipšico konstantos įvertis turi būti atnaujintas kiekvienai tikslo funkcijai atskirai. Surogatiniai Lipšico rėžiai yra sudaromi naudojant tik vieną viršūnę su geriausia funkcijos reikšme (žr. (37)) ir (38) įvertis yra naudojamas, kad būtų randama $\boldsymbol{R}(\cdot)$ (3.3) reikšmė analitiškai.

Paieškos globalumas gali būti pasirinktas nustatant Libre parametro $\alpha \in [0, \infty)$ reikšmę. Vieno kriterijaus algoritmo versijoje du kriterijai (5) yra naudojami parinkti, kurie simpleksai turėtų būti dalinami. Pirmasis kriterijus yra didžiausias tikėtinas turimo sprendinio pagerėjimas, kuris vieno kriterijaus atveju yra mažiausia rasta funkcijos reikšmė f_{min} ir daugiakriteriniu atveju tai yra Pareto aibės aproksimacija Y^{Pareto}. Antrasis kriterijus yra simplekso diametras $\Delta(\cdot)$. Jeigu daugiakriteriniu atveju didžiausio tikėtino turimo sprendinio pagerėjimo įvertinimui būtų naudojamas (52) apibrėžimas,

Algorithm 4: Daugiakriterinės Libre algoritmo versijos aprašymas

Normalizuoti leistinąją sritį D į vienetinį hiperkubą $\overline{D}$.
Padengti $\overline{D}$ nepersidengiančiais simpleksais $S = \{S_i : \overline{D} = \cup S_i,\ i = 1, ..., d!\}$ naudojant kombinatorinį viršūnių trianguliavimo algoritmą.
Įvertinti $\{\boldsymbol{f}(\boldsymbol{v}_i) : \boldsymbol{v}_i$ unikalios S viršūnės, $i = 1, ..., 2^d\}$. Nustatyti $m = 2^d$, $\widetilde{\boldsymbol{L}} = 0$.
Rasti Pareto fronto aproksimaciją Y^{Pareto}.
while *sustojimo kriterijus nėra patenkintas* **and** $m < M_{max}$ **do**
  for $k = 1$ ***to*** $|S|$ **do**
    rasti $\widetilde{\boldsymbol{L}}_k = (\widetilde{L}_k^1, ..., \widetilde{L}_k^n)$, $\widetilde{L}_k^i = \max\left\{\frac{|f^i(\boldsymbol{v}_i) - f^i(\boldsymbol{v}_j)|}{||\boldsymbol{v}_i - \boldsymbol{v}_j||} : \boldsymbol{v}_i, \boldsymbol{v}_j \in V(S_k), \boldsymbol{v}_i \neq \boldsymbol{v}_j\right\}$.
  Atnaujinti $\widetilde{\boldsymbol{L}} = \left(\max_{k\in\{1,...,|S|\}} \widetilde{L}_k^1, ..., \max_{k\in\{1,...,|S|\}} \widetilde{L}_k^n\right)$.
  for $k = 1$ ***to*** $|S|$ **do**
    rasti $T_k = T(S_k, Y^{Pareto}, \widetilde{\boldsymbol{L}}\alpha)$ remiantis (52), čia (37), (38) yra naudojami rasti Lipšico rėžiams formulėje (3.3) ir $\widetilde{\boldsymbol{L}}\alpha$ yra naudojamas vietoje $\boldsymbol{L}$.
  Nustatyti aibę simpleksų dalinimui:
    $P = \{S_i : S_i \in S,\ S_i$ - ikilas Pareto optimalus (53) sprendinys$\}$.
  foreach $S_k \in P$ **do**
    Padalinti S_k į du naujus simpleksus S_k^1, S_k^2, pridedant viršūnę $\boldsymbol{v}$ ilgiausios S_k briaunos viduryje.
    Atnaujinti $S = S\backslash\{S_k\} \cup \{S_k^1, S_k^2\}$. Jei $\boldsymbol{v}$ yra nauja viršūnė, įvertinti $\boldsymbol{f}(\boldsymbol{v})$, nustatyti $m = m + 1$ ir atnaujinti Pareto aibės aproksimaciją Y^{Pareto}.

tada simpleksų aibė dalinimui gali būti parenkama išsprendžiant:

$$\max_{S_i \in S}(T(S_i, Y^{Pareto}, \alpha\widetilde{\boldsymbol{L}}),\ \Delta(S_i)), \tag{53}$$

čia $\widetilde{\boldsymbol{L}}$ yra Lipšico konstantos įverčių vektorius. Buvo parodyta [11], kad vieno kriterijaus algoritmo versijoje didesnė α reikšmė sąlygoja globalesnę paiešką. Didesnė α reikšmė padaro, kad tik simpleksai, turintys didesnį diametrą, būtų parenkami dalinimui. Tas pats α efektas išlieka ir daugiakriteriniu atveju.

Daugiakriterinė Libre algoritmo versija turi sekti ir nuolatos atnaujinti Pareto fronto aproksimaciją Y^{Pareto}. Tai gali būti pasiekiama pirmiausia inicializuojant $Y^{Pareto} = \varnothing$ ir po kiekvieno naujo funkcijos įvertinimo atnaujinant Y^{Pareto}, į aibę įterpiant nedominuojamus taškus ir iš jos išmetant dominuojamus, jeigu tokių atsiranda įterpus naują tašką.

3.4.2 Eksperimentinis tyrimas

3.4.2.1 Veikimo palyginimas su NSGA-II

Daugiakriterinis Libre algoritmas buvo eksperimentiškai įvertintas naudojant du populiarius daugiakriterinių testinių uždavinių rinkius ZDT [45] ir DTLZ [6]. Rezultatai buvo palyginti su vienu iš populiariausių genetinių algoritmų NSGA-II [5].

Eksperimentams buvo naudojama NSGA-II algoritmo realizacija iš DEAP karkaso [8] ir buvo naudojamos numatytosios parametrų reikšmės. Libre algoritmo $\alpha = 0.1$ reikšmė buvo naudojama. Uždavinių dimensija buvo nustatyta į 4 (tokią pačią kaip ir 5'tos ir 6'tos GKLS testinių uždavinių klasės). Algoritmai buvo vykdomi, kol buvo išnaudojamas fiksuotas 15000 tikslo funkcijų įvertinimų biudžetas. Buvo matuojama hyper tūrio reikšmės kaita (žr. 5a-5l paveikslus), kur X ir Y ašys rodo tikslo funkcijos įvertinimų skaičių ir hyper tūrio reikšmę atitinkamai. Mėlyna linija žymi Libre algoritmo rezultatus, o žalias plotas žymi (nuo mažiausios iki didžiausios) reikšmes, gautas su 10 skirtingų NSGA-II įvykdymų.

Rezultatai rodo, kad pasiūlytasis daugiakriterinis Libre algoritmas veikia geriau (pasiekia geriausią rastą hyper tūrio reikšmę greičiau) nei NSGA-II 7 atvejais iš 12. Galima daryti išvadą, kad su daugiakriteriniu algoritmu, kuris buvo gautas pritaikius pasiūlytąją apibendrinimo strategiją, buvo gautas panašus efektyvumas į NSGA-II algoritmo, taigi tiek pasiūlytoji strategija, tiek Libre algoritmas turėtų būti toliau tiriami.

3.4.2.2 Veikimo palyginimas su Lipšicinio optimizavimo algoritmais

Libre algoritmas buvo palygintas su OSWCO (vieno žingsnio blogiausio scenarijaus optimalus algoritmas) [44] ir NUC (netolygaus padengimo metodas) [7] algoritmais. Tie patys du dvikriteriniai uždaviniai kaip ir tos pačios skaitinės charakteristikos kaip ir [7, 44] buvo naudojamos eksperimentuose. Charakteristikos hipertūris ir Pareto fronto taškų išsidėstymo tolygumas buvo pasiūlytos [46].

Pirmosios dvikriterinės testinės problemos (PE1) apibrėžimas:

$$\min_{x\in[0,2]^2} \boldsymbol{f}(x), \quad f^1(x) = x_1, \quad f^2(x) = \min(|x_1 - 1|, 1.5 - x_1) + x_2 + 1. \tag{54}$$

8 lentelė: Algoritmų veikimo palyginimas su dviem dvikriteriniais uždaviniais

Uždavinys	Algoritmas	Iškvietimų	$\lvert Y^{Pareto}\rvert$	Hipertūris	Tolygumas	Nadir
PE1	LIBRE $\alpha = 0.1$	438	**163**	**3.61328**	-	(2,3)
	OSWCO	435	92	3.60519	-	
	NUC	490	36	3.42	-	
PE2	LIBRE $\alpha = 0.1$	484	**221**	**0.325086**	**0.086283**	(1,1)
	OSWCO	498	68	0.308484	0.174558	
	NUC	515	29	0.306	0.210	

Šios problemos Pareto frontas turi trūkį, todėl tolydumo charakteristika nėra tinkama šiam uždaviniui.

Antrojo dvikriterinio uždavinio (PE2) apibrėžimas:

$$\min_{x\in[0,1]^2} \boldsymbol{f}(x), \quad f^1(x) = (x_1 - 1)x_2^2 + 1, \quad f^2(x) = x_2. \tag{55}$$

Skaitiniai rezultatai pateikti 8 lentelėje. Eksperimentuose buvo naudojama LIBRE parametro $\alpha = 0.1$ reikšmė. Kaip galima matyti iš 8 lentelės, LIBRE algoritmo rezultatai yra geresni nei alternatyvų.

4 Rezultatai ir išvados

Buvo pasiūlytas algoritmas, kuris kombinuoja elementus iš DISIMPL-V algoritmo ir adaptyvaus Lipšico konstantos įverčio strategijas. Konkrečiau, padengimas simpleksais, posričių parinkimo strategija ir surogatinių Lipšico rėžių apskaičiavimas buvo perimti iš originalaus DISIMPL-V algoritmo. Pasiūlytasis algoritmas buvo eksperimentiškai palygintas su kitais DIRECT tipo Lipšico optimizavimo algoritmais naudojant 800 tikslo funkcijų, sugeneruotų su GKLS testinių funkcijų generatoriumi. Atlikta algoritmo vienintelio parametro α, reguliuojančio paieškos globalumą, eksperimentinė analizė.

Apsvarstytos keturios strategijos sudaryti surogatinius Lipšico rėžius simplekse. Jos buvo pritaikytos pasiūlytajam LIBRE algoritmui ir eksperimentiškai parinkta efektyviausia.

Pasiūlyta strategija apibendrinti vieno kriterijaus Lipšico optimizavimo algoritmams daugiakriteriniams uždaviniams. Ši strategija yra grįsta potencialaus Pareto fronto aproksimacijos pagerėjimo, kuris galėtų būti gaunamas padalinant simpleksą, įverčiu. Ši strategija buvo pritaikyta Libre algoritmui ir gautasis daugiakriterinis algoritmas buvo eksperimentiškai palygintas su NSGA-II algoritmu.

Išvados:

1. Eksperimentiškai parodyta, kad pasiūlytasis Libre algoritmas veikia geriausiai iš nagrinėtų alternatyvų sudėtingų testinių funkcijų klasių blogiausių scenarijų atvejais. Blogiausių scenarijų rezultatai Libre algoritmui buvo ne geriausi tik 2 klasėms (iš paprastų funkcijų) iš nagrinėtų 8. Be to, su Libre algoritmu gauti rezultatai buvo geresni nei Dsimpl-v algoritmo 7 klasėms iš 8 tiek blogiausio, tiek vidutinio scenarijų atvejais.

2. Eksperimentų su Libre algoritmu rezultatai parodė, kad efektyviausia (tikslo funkcijos įvertinimų skaičiaus atžvilgiu) strategija įvertinti surogatiniams Lipšico rėžiams simplekse yra v_{min}, kai rėžiai sudaromi naudojant tik vieną viršūnę su mažiausia funkcijos reikšme. Naudojant šią strategiją gaunama geriausia vidutinė funkcijų įvertinimų trukmė ir mažiausias tikslo funkcijos įvertinimų skaičius su visomis nagrinėtomis testinių funkcijų klasėmis. Rėžiai, gaunami su šia strategija, yra mažiausiai griežti iš visų nagrinėtų alternatyvų, todėl galima daryti išvadą, kad griežtesnių surogatinių Lipšico rėžių naudojimas nebūtinai sąlygoja efektyvumo padidėjimą Direct tipo algoritmuose.

3. Lyginamieji eksperimentai atskleidė, kad apibendrinto daugiakriteriniam atvejui Libre algoritmo versijos efektyvumas yra panašus į NSGA-II sprendžiant žemo dimensiškumo optimizavimo uždavinius. 7 iš 12 daugiakriteriniams uždaviniams iš ZDT ir DTLZ testinių uždavinių rinkinių pasiūlytasis algoritmas pasiekė ne blogesnes hipertūrio reikšmes bet kokiam funkcijų įvertinimų skaičiui.

Literatūra

[1] C. A. Baker, L. T. Watson, B. Grossman, W. H. Mason ir R. T. Haftka. “Parallel global aircraft configuration design space exploration”. (2000). (Preprint).

[2] A. R. Butz. "Space filling curves and mathematical programming". *Information and Control* 12.4 (1968), puslapiai 314–330.

[3] Yu. M. Danilin. "Estimation of the efficiency of an absolute-minimum-finding algorithm". *USSR Computational Mathematics and Mathematical Physics* 11.4 (1971), puslapiai 261–267.

[4] Yu. M. Danilin ir S. A. Pijavskij. "An algorithm for finding the absolute minimum". *Theory of Optimal Decisions* 2 (1967), puslapiai 25–37.

[5] K. Deb, A. Pratap, S. Agarwal ir T. Meyarivan. "A fast and elitist multiobjective genetic algorithm: NSGA-II". *IEEE transactions on evolutionary computation* 6.2 (2002), puslapiai 182–197.

[6] K. Deb, L. Thiele, M. Laumanns ir E. Zitzler. "Scalable multi-objective optimization test problems". *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*. Tomas 1. IEEE. 2002, puslapiai 825–830.

[7] Yu. G. Evtushenko ir M. A. Posypkin. "Nonuniform covering method as applied to multicriteria optimization problems with guaranteed accuracy". *Computational Mathematics and Mathematical Physics* 53.2 (2013), puslapiai 144–157.

[8] F. A. Fortin, F. M. De Rainville, M. A. Gardner, M. Parizeau ir C. Gagné. "DEAP: Evolutionary algorithms made easy". *Journal of Machine Learning Research* 13.Jul (2012), puslapiai 2171–2175.

[9] J. M. Gablonsky ir C. T. Kelley. "A locally-biased form of the DIRECT algorithm". *Journal of Global Optimization* 21.1 (2001), puslapiai 27–37.

[10] M. Gaviano, D. E. Kvasov, D. Lera ir Ya. D. Sergeyev. "Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization". *ACM Transactions on Mathematical Software (TOMS)* 29.4 (2003), puslapiai 469–480.

[11] A. Gimbutas ir A. Žilinskas. "An algorithm of simplicial Lipschitz optimization with the bi-criteria selection of simplices for the bi-section". *Journal of Global Optimization* 71.1 (2018), puslapiai 115–127.

[12] Albertas Gimbutas. "Globalios optimizacijos algoritmas, naudojantis lokalų Lipšico konstantos įvertį". *Jaunųjų mokslininkų darbai* 1.45 (2016), puslapiai 47–53.

[13] R. Grbić, E. K. Nyarko ir R. Scitovski. "A modification of the DIRECT method for Lipschitz global optimization for a symmetric function". *Journal of Global Optimization* 57.4 (2013), puslapiai 1193–1212.

[14] P. Hansen ir B. Jaumard. "Lipshitz optimization". *Handbook of Global Optimization*. Redagavo R. Horst ir P. Pardalos. Dodrecht: Kluwer Academic Publisher, 1995, puslapiai 407–493.

[15] R. Horst. "A general class of branch-and-bound methods in global optimization with some new approaches for concave minimization". *Journal of Optimization Theory and Applications* 51.2 (1986), puslapiai 271–291.

[16] R. Horst ir H. Tuy. *Global Optimization: Deterministic Approaches*. Springer Science & Business Media, 2013.

[17] V. V. Ivanov. "Optimal Algorithms of Minimization in the Class of Functions with the Lipschitz Condition." *Information Processing* 71 (1972), puslapiai 1324–1327.

[18] B. Jaumard, H. Ribault ir T. Herrmann. "An on-line cone intersection algorithm for global optimization of multivariate Lipschitz functions". *Les Cahiers du GERAD* (1995).

[19] D. R. Jones, C. D. Perttunen ir B. E. Stuckman. "Lipschitzian optimization without the Lipschitz constant". *Journal of Optimization Theory and Applications* 79.1 (1993), puslapiai 157–181.

[20] D. E. Kvasov, C. Pizzuti ir Ya. D. Sergeyev. "Local tuning and partition strategies for diagonal GO methods". *Numerische Mathematik* 94.1 (2003), puslapiai 93–106.

[21] D. E. Kvasov ir Ya. D. Sergeyev. "A univariate global search working with a set of Lipschitz constants for the first derivative". *Optimization Letters* 3.2 (2009), puslapiai 303–318.

[22] D. E. Kvasov ir Ya. D. Sergeyev. "Lipschitz gradients for global optimization in a one-point-based partitioning scheme". *Journal of Computational and Applied Mathematics* 236.16 (2012), puslapiai 4042–4054.

[23] Q. Liu ir W. Cheng. "A modified DIRECT algorithm with bilevel partition". *Journal of Global Optimization* 60.3 (2014), puslapiai 483–499.

[24] Q. Liu ir J. Zeng. "Global optimization by multilevel partition". *Journal of Global Optimization* 61.1 (2015), puslapiai 47–69.

[25] R. H. Mladineo. "An algorithm for finding the global maximum of a multimodal, multivariate function". *Mathematical Programming* 34.2 (1986), puslapiai 188–200.

[26] J. Mockus. "On the Pareto optimality in the context of Lipschitzian optimization". *Informatica* 22.4 (2011), puslapiai 521–536.

[27] R. Paulavičius, Ya. D. Sergeyev, D. E. Kvasov ir J. Žilinskas. "Globally-biased DISIMPL algorithm for expensive global optimization". *Journal of Global Optimization* 59.2-3 (2014), puslapiai 545–567.

[28] R. Paulavičius ir J. Žilinskas. *Simplicial global optimization*. Springer, 2014.

[29] R. Paulavičius ir J. Žilinskas. "Simplicial Lipschitz optimization without the Lipschitz constant". *Journal of Global Optimization* 59.1 (2014), puslapiai 23–40.

[30] S. Pijavskij. "An algorithm for finding the global extremum of function". *Optimal Decisions* 2 (1967), puslapiai 13–24.

[31] J. D. Pintér. *Global optimization in action: continuous and Lipschitz optimization: algorithms, implementations and applications*. Tomas 6. Nonconvex Optimization and Its Applications. Kluwer Academic Publishers, 1996.

[32] Ya. D. Sergeyev. "An information global optimization algorithm with local tuning". *SIAM Journal on Optimization* 5.4 (1995), puslapiai 858–870.

[33] Ya. D. Sergeyev. "Efficient strategy for adaptive partition of N-dimensional intervals in the framework of diagonal algorithms". *Journal of Optimization Theory and Applications* 107.1 (2000), puslapiai 145–168.

[34] Ya. D. Sergeyev ir D. E. Kvasov. *Deterministic Global Optimization: An Introduction to the Diagonal Approach*. Springer, 2017.

[35] Ya. D. Sergeyev ir D. E. Kvasov. *Diagonalnyje metody globalnoj optimizacii*. In Russian. Moscow: Fizmatlit, 2008.

[36] Ya. D. Sergeyev ir D. E. Kvasov. "Global search based on efficient diagonal partitions and a set of Lipschitz constants". *SIAM Journal on Optimization* 16.3 (2006), puslapiai 910–937. DOI: http://dx.doi.org/10.1137/040621132.

[37] Ya. D. Sergeyev ir D. E. Kvasov. "Global search based on efficient diagonal partitions and a set of Lipschitz constants". *SIAM Journal on Optimization* 16.3 (2006), puslapiai 910–937.

[38] Ya. D. Sergeyev, M. S. Mukhametzhanov, D. E. Kvasov ir D. Lera. "Derivative-free local tuning and local improvement techniques embedded in the univariate global optimization". *Journal of Optimization Theory and Applications* 171.1 (2016), puslapiai 186–208.

[39] B. O. Shubert. "A sequential method seeking the global maximum of a function". *SIAM Journal on Numerical Analysis* 9.3 (1972), puslapiai 379–388.

[40] R. G. Strongin. "Algorithms for multi-extremal mathematical programming problems employing the set of joint space-filling curves". *Journal of Global Optimization* 2.4 (1992), puslapiai 357–378.

[41] R. G. Strongin ir Ya. D. Sergeyev. *Global optimization with non-convex constraints: Sequential and parallel algorithms*. Kluwer Academic Publishers, Dordrecht, 2000.

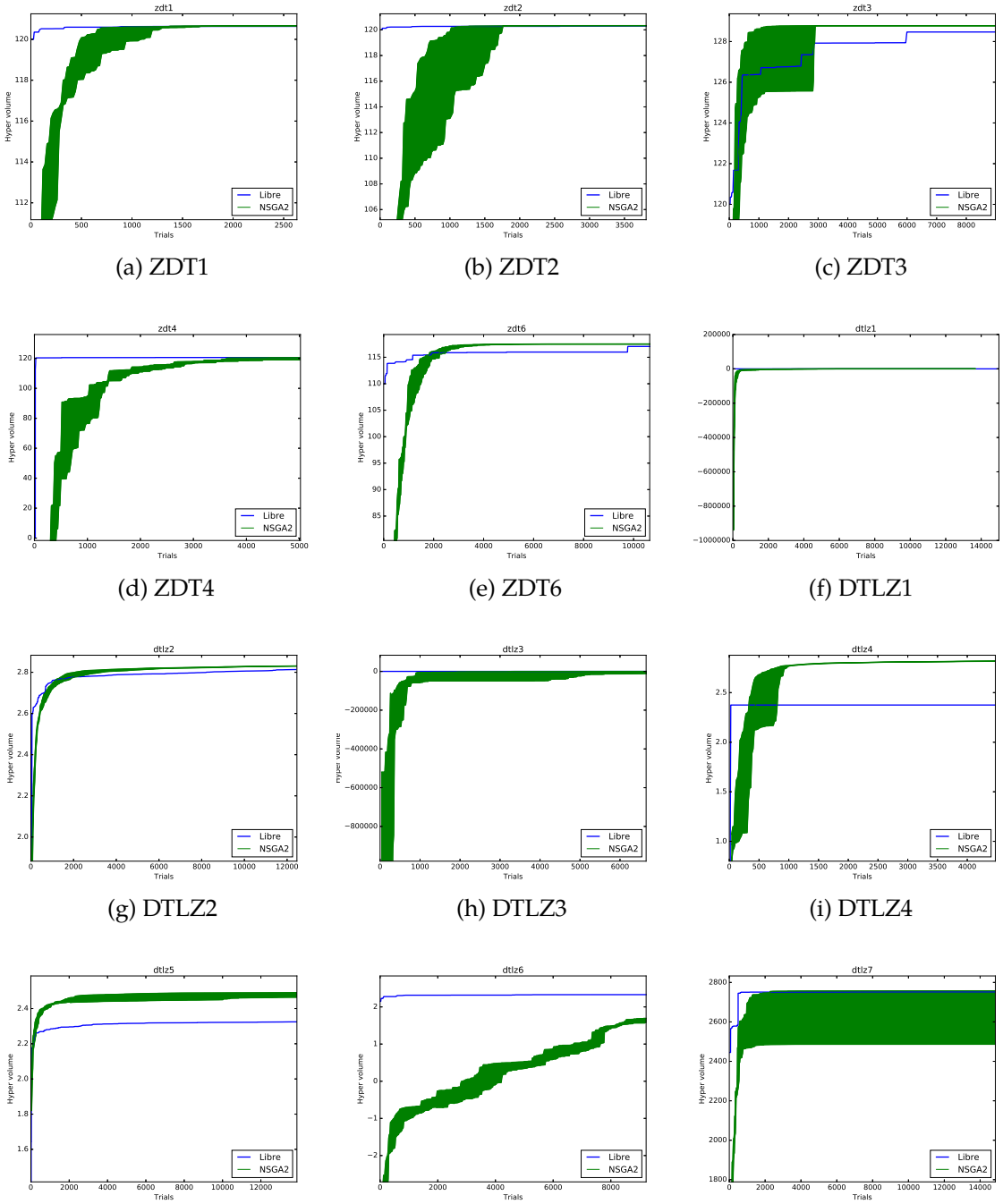
[42] A. Žilinskas. "A one-step worst-case optimal algorithm for bi-objective univariate optimization". *Optimization Letters* 8.7 (2014), puslapiai 1945–1960.

[43] A. Žilinskas ir G. Gimbutienė. "On one-step worst-case optimal trisection in univariate bi-objective Lipschitz optimization". *Communications in Nonlinear Science and Numerical Simulation* 35 (2016), puslapiai 123–136.

[44] A. Žilinskas ir J. Žilinskas. "Adaptation of a one-step worst-case optimal univariate algorithm of bi-objective Lipschitz optimization to multidimensional problems". *Communications in Nonlinear Science and Numerical Simulation* 21.1-3 (2015), puslapiai 89–98.

[45] E. Zitzler, K. Deb ir L. Thiele. "Comparison of multiobjective evolutionary algorithms: empirical results". *Evolutionary computation* 8.2 (2000), puslapiai 173–195.

[46] E. Zitzler, J. Knowles ir L. Thiele. "Quality assessment of Pareto set approximations". *Multiobjective Optimization*. Springer, 2008, puslapiai 373–404.



(a) ZDT1 (b) ZDT2 (c) ZDT3

(d) ZDT4 (e) ZDT6 (f) DTLZ1

(g) DTLZ2 (h) DTLZ3 (i) DTLZ4

(j) DTLZ5 (k) DTLZ6 (l) DTLZ7

5 pav.: Comparison of Libre and NSGA-II [5] algorithms using continuous test from problems from ZDT and DTLZ test problem sets

Doktoranto publikacijos disertacijos tema

Publications in peer reviewed periodicals

1. Gimbutas, A. (2016). Globalios optimizacijos algoritmas, naudojantis lokalų Lipšico konstantos įvertį. *Jaunųjų mokslininkų darbai*, Vol. 1, No. 45, pp. 47-53. doi:10.21277/jmd.v1i45.44

2. Gimbutas, A., and Žilinskas, A. (2017). An algorithm of simplicial Lipschitz optimization with the bi-criteria selection of simplices for the bi-section. *Journal of Global Optimization*, pp. 1-13. doi:10.1007/s10898-017-0550-9

Peer reviewed conference publications

3. Gimbutas, A. and Žilinskas, A. (2016). On global optimization using an estimate of Lipschitz constant and simplicial partition. In: *Numerical Computations: Theory and Algorithms*. Calabria: AIP Publishing, Vol. 1776, No. 1, p.060012. doi:10.1063/1.4965346

4. Gimbutas, A. and Žilinskas, A. (2018). Generalization of Lipschitzian global optimization algorithms to the multi-objective case. In: *International Workshop on Optimization and Learning: Challenges and Applications*. Alicante, pp.36-37.

Other presentations in conferences

1. Gimbutas, A. (2014). One-step worst-case optimal bivariate algorithm for bi-objective optimization. *Data Analysis Methods for Software Systems*. Druskininkai.

2. Gimbutas, A. (2015). Daugiadimensinis globalios optimizacijos algoritmas naudojantis adaptyviąją Lipšico konstantą. In: *Computer Days*. Panevėžys.

3. Gimbutas, A. (2015). DAKIS - algoritmų įvertinimo ir palyginimo įrankis. In: *Operacijų tyrimas ir taikymai*. Panevėžys.

4. Gimbutas, A. (2015). Multicriteria Lipschitz Optimization Algorithm Using Local Lipschitz Constant Estimate. *Data Analysis Methods for Software Systems*. Druskininkai, p. 20.

5. Gimbutas, A. (2016). Daugiakriterė globali optimizacija naudojant adaptyvią Lipšico konstantą. In: *Operacijų tyrimas ir taikymai*. Kaunas.

6. Gimbutas, A. and Žilinskas (2016). Remarks on a Multi-Criteria Simplicial Optimization with an Estimate of Lipschitz constant. *Data Analysis Methods for Software Systems*. Druskininkai, p. 22.

7. Gimbutas, A. (2017). Daugiaobjektinis Lipšico simpleksinis optimizavimas su Lipšico konstantos įvertinimu. In: *Computer Days*. Kaunas, p. 25.

Albertas Gimbutas

NEIŠKILIOJO OPTIMIZAVIMO ALGORITMAS SU NAUJU BIKRITERINIU POTENCIALIŲJŲ SIMPLEKSŲ IŠRINKIMU NAUDOJANT LIPŠICO KONSTANTOS ĮVERTĮ

Daktaro disertacijos santrauka

Fiziniai mokslai

Informatika (09P)

Redaguota „Vertimų karaliai“, MB

Albertas Gimbutas

NONCONVEX OPTIMIZATION ALGORITHM WITH A NEW BI-CRITERIA SELECTION OF POTENTIAL SIMPLICES USING AN ESTIMATE OF LIPSCHITZ CONSTANT

Summary of Doctoral Dissertation

Physical Sciences

Informatics (09P)

Editor Nijolė Požėraitytė